Choose the BEST answer of those given and enter your choice on the Answer Sheet. You may choose multiple options, but the point value will be divided by the number of options chosen.

#1 In most C-Based languages, which is the following order of precedence (from highest to lowest):
(a) postfix increment/decrement; unary subtraction; multiplication; binary addition.
(b) multiplication/division; unary addition/subtraction; binary addition/subtraction.
(c) exponentiation; division; subtraction.
(d) binary subtraction; remainder; unary addition; exponentiation.
(e) multiplication; division; addition; subtraction.

#2 When two operators of the same precedence are adjacent in an expression, evaluation is performed
(a) left to right.
(b) right to left.
(c) according to the order of precedence of the operators.
(d) using the higher priority operator first.
(e) according to the associativity of the operators.

#3 What is a "side effect"?
(a) When the value of an expression must be delayed until another expression is evaluated.
(b) When the evaluation of an expression causes unintended consequences.
(c) When the order of evaluation of function's parameters affects the value of the expression.
(d) When the evaluation of an expression changes the value of a variable.
(e) When an expression cannot be fully evaluated until after all of its parameters are evaluated.

#4 Which of the following statements is NOT true regarding operator overloading:
(a) Operator overloading can harm the readability of a program.
(b) Operator overloading is resolved based on the types of the operands seen by the operator.
(c) Operator overloading can improve the readability of a program.
(d) Most languages inherently overload some operators.
(e) Overloaded operators always operate on user-defined data types.

#5 In C, if a=6, b=4, and c=2, the result of evaluating a < b < c will be
(a) -1
(b) True
(c) 1
(d) False
(e) 0

#6 An assignment statement with a conditional target means that

(a) the variable that a value is assigned to depends on the evaluation of an expression.

(b) the value of an expression may be assigned to multiple variables.

(c) the value that is assigned to a variable depends on the evaluation of an expression.

(d) the target may or may not be written to depending on the value of the expression to be written.

(e) the expression being evaluated depends on which variable the value will be assigned to.

#7 The ambiguous-else (or dangling-else) problem is generally resolved by what rule?

(a) each 'else' clause is paired with the most recent 'then' clause.

(b) each 'else' clause is paired with the most recent 'if' expression.

(c) each 'else' clause is paired with the most recent unpaired 'if' expression.

(d) each 'else' clause is matched to the most recent 'if' expression at the same level of indentation.

(e) each 'else' clause must be written so as to be unambiguous.

#8 The multiple selection statement is commonly known as

(a) the decision-based execution structure.

(b) the if-then-else structure.

(c) the controlled-execution structure.

(d) the if-goto structure.

(e) the switch structure.

#9 When control flows from one case label to the next, this is known as

(a) multi-case execution.

(b) a syntax error.

(c) a logic error.

(d) case ambiguity.

(e) flow-through.

#10 In C, if a switch expression does not have a 'default' case, then

(a) the first case segment is executed by default.

(b) the last case segment is executed by default.

(c) a compile-time error results.

(d) the entire construct is skipped if none of the case labels match the expression.

(e) a run-time exception is thrown.

#11 Iterative statements are broadly categorized as belong to one of which two categories?
(a) Internally-controlled and externally-controlled.
(b) Sentinel-controlled and Logically controlled.
(c) Counter-controlled and Logically controlled.
(d) Fixed-pattern and flexible-pattern.
(e) mutable and immutable.

#12 In C, which of the following is true of the looping constructs?
(a) All of looping constructs are required in order to have a complete language.
(b) Any loop that can be written using one construct can be implemented using either of the remaining constructs.
(c) The for loop can only be used for counter-controlled loops.
(d) The loop body can contain either 'break' or 'continue' statements, but never both.
(e) The do loop forces execution of the loop body and this cannot be accomplished using a while loop.

#13 With the C 'for' statement, which of the following statements is NOT true?
(a) All three of the expressions in the header are optional and could all three be left out.
(b) The 'for' statement is essentially a more formalized implementation of the 'while' statement.
(c) It is legal to branch into the middle of the loop body.
(d) The value of the loop variable can be changed within the body of the loop code.
(e) An infinite loop results if the 'for' header's the test expression (the middle expression) is left out.

#14 Most modern languages still include a 'goto' statement (or equivalent) for what reason?
(a) They don't -- modern languages almost always leave out the 'goto' statement entirely.
(b) It it not possible to implement a truly general purpose language without it.
(c) Eliminating it would break at least some previously written code.
(d) There are situations in which the judicious use of a 'goto' statement is reasonable and remains readable.
(e) The language designers usually choose to accommodate "traditional" programming styles.

#15 In addition to sequences, what two control structures are absolutely required to express computation?
(a) Pre-test loops and post-test loops.
(b) Selection and pre-test loops.
(c) Selection and post-test loops.
(d) Goto statements and selection statements.
(e) Selection statements and counter-controlled loops.

#16 Data-based control structures designed to execute the loop body one time for each data element are generally known as
(a) logic-controlled.
(b) data-controlled.
(c) collections.
(d) iterators.
(e) counter-controlled.

#17 In many processors, particularly simple ones, all of the control structures in a language are ultimately implemented using
(a) loop and branch instruction
(b) selection and iteration instructions.
(c) 'if' and 'while' instructions.
(d) stack-based branch instructions.
(e) conditional and unconditional branch (i.e., goto) instructions.

#18 Why do most languages include more control statements than are absolutely required for completeness?
(a) to eliminate ambiguity.
(b) backwards compatability.
(c) for simplicity.
(d) Readability and writability.
(e) consistency with common usage.

#19 The method used by some languages to eliminate the ambiguous-if problem syntactically is to require
(a) that any nested 'if' clause only follow unpaired 'else' clauses.
(b) that all selection statements have both a 'then' and an 'else' clause.
(c) the use of a reserved word to close the selection entity explicitly.
(d) that all 'else' clauses be compound statements.
(e) that any nested 'if' clause only follow paired 'else' clauses.

#20 Except for coroutines, subprograms in modern languages generally have all of the following characteristics except:
(a) control always returns to the caller when subprogram execution terminates.
(b) a single entry point.
(c) the calling program is suspended during execution of the called routine.
(d) only one subprogram is active at any given time.
(e) a single exit point.

#21 The general distinction between a "procedure" and a "function" is that
- (a) procedures have access to all local variables of the caller.
- (b) procedures do not take any arguments.
- (c) functions return values while procedures do not.
- (d) procedures can return multiple values while functions can only return a single (though possibly compound) value.
- (e) functions are not allowed to have side effects.

#22 In object-oriented languages, a "method" generally refers to a subprogram that is
- (a) associated with a class but need not be associated with an instance of that class.
- (b) equivalent to a procedure in non objected-oriented languages.
- (c) callable from anywhere in a program and is not associated with any particular class.
- (d) equivalent to a function in non objected-oriented languages.
- (e) called from an instance of class and has access to all of that instance's data.

#23 A generic subprogram is one whose
- (a) calling environment includes all variables in all of its dynamic parents.
- (b) computation can be done on data of different types in different calls.
- (c) calling environment includes all variables in all of its static parents.
- (d) computation solves a general problem instead of a specific problem.
- (e) parameters are typeless.

#24 A 'closure' is a
- (a) means of hiding information from the caller of a nested program.
- (b) call to a nested subprogram that includes the necessary referencing environment.
- (c) data structure that tracks the the referencing environment of one or more nested subprograms.
- (d) nested subprogram combined with its referencing environment.
- (e) set of subprograms that can all call each other.

#25 Which parameter passing scheme is also known as pass-by-copy?
- (a) Pass-by-name
- (b) Pass-by-result
- (c) Pass-by-reference
- (d) Pass-by-value-result
- (e) Pass-by-value

#26 Which of the following parameter passing schemes gives gives rise to issues associated with aliases?
(a) pass-by-name
(b) pass-by-value
(c) pass-by-result
(d) pass-by-reference
(e) pass-by-value-result

#27 Because C uses row-major array indexing, which of the following is a workable header for passing an 20-row, 10-col array to a function?
(a) void fun(int matrix[10][20]) {...}
(b) void fun(int matrix[][10]) {...}
(c) void fun(int matrix[][20]) {...}
(d) void fun(int matrix[20][]) {...}
(e) void fun(int matrix[10][]) {...}

#28 The method of establishing the referencing environment for nested subprograms in statically-scoped languages is almost always
(a) shallow binding.
(b) lexical binding.
(c) referential binding.
(d) deep binding.
(e) ad hoc binding.

#29 An activation record is a data structure that holds all but which of the following information about a function call?
(a) Static local variables.
(b) Dynamic local variables.
(c) Parameters.
(d) Return addresses.
(e) Dynamic Links.

#30 In an activation record, the pointer to the base of the activation record of the caller is known as
(a) the return address.
(b) a static link.
(c) a dynamic link.
(d) the parent link.
(e) the child link.

#31 In an activation record, the pointer to the base of the activation record for the most recently called lexical parent of the called program is known as

(a) a static link.
(b) the return address.
(c) the parent link.
(d) a dynamic link.
(e) the child link.

#32 What is the only difference, in terms of the activation record instances, when a subprogram makes a recursive call to itself?

(a) The dynamic link will be self referential.
(b) There is no difference.
(c) The parent link will be self referential.
(d) The child link will be self referential.
(e) The static link will be self referential.

#33 The dynamic link, in static-scoped languages, is used to

(a) access the nonlocal referencing environment in the case of a closure.
(b) access nonlocal variables.
(c) provide traceback information when a run-time error occurs.
(d) return program control to the caller when the subprogram terminates.
(e) provide access to global variables.

#34 The dynamic link, in dynamic-scoped languages, is used to

(a) access nonlocal variables.
(b) provide traceback information when a run-time error occurs.
(c) access the nonlocal referencing environment in the case of a closure.
(d) return program control to the caller when the subprogram terminates.
(e) provide access to global variables.

#35 Which element is not needed in an activation record when subprograms cannot be nested?

(a) A dynamic link.
(b) Local variables.
(c) Parameters.
(d) The return address.
(e) A static link.

#36 Blocks (compound statements) that provide user-specified local scopes can be implemented as
(a) anonymous functions.
(b) parameterless nested subprograms, even in languages that don't support nested subprograms.
(c) auto-generated subprograms.
(d) closures.
(e) paramaterless nested subprograms, provided the language supports nested subprograms.

#37 Blocks within a subprogram can be implemented
(a) by including any block variables as part of the subprogram's set of local variables on the stack.
(b) as auto-generated subprograms.
(c) paramaterless nested subprograms, provided the language supports nested subprograms.
(d) as closures.
(e) as anonymous functions.

#38 Dynamic scoping that is implemented via tracing through the activation records for the nearest instance of a variable name is known as
(a) static access.
(b) dynamic access.
(c) deep access.
(d) stack access.
(e) shallow access.

#39 Dynamic scoping that is implemented via maintaining a separate stack for each named variable is known as
(a) shallow access.
(b) deep access.
(c) stack access.
(d) static access.
(e) dynamic access.

#40 A "central table" is one alternative method of implementing
(a) stack access.
(b) shallow access.
(c) static access.
(d) deep access.
(e) dynamic access.

#41 In some processors, particularly RISC processors, subprogram parameters are not passed on the stack but, instead, via
(a) static memory locations.
(b) linked lists maintained by the processor.
(c) a central table.
(d) processor registers.
(e) a separate parameter stack.

#42 A subprogram is 'active'
(a) from the time it is called until the time it terminates.
(b) whenever its static parent is active.
(c) only when its activation record is on top of the stack.
(d) from the time it is called until the main program terminates.
(e) only when its code is being executed.

#43 Which of the following operators is usually NOT right associative?
(a) The address dereference operator (in C: '*****').
(b) The pre-increment operator (in C: '**++**').
(c) The post-decrement operator (in C: '**--**').
(d) The bitwise negation (in C: '**~**').
(e) The unary minus operator (in C: '**-**').

#44 What value is output by the following C program?

```
#include <stdio.h>
int main(void)
{
    int a = 5, b = 10, c = -1, d;

    if ( (a-- < 10) && (b-- > 10) && (c++) )
       d = (a + 2*b)*c;
    else
       d = (a - 2*b)*c;

    printf("%i\n", d);
    return 0;
}
```

(a) -24.
(b) -22.
(c) 0.
(d) 14.
(e) 16.

#45 Which of the following is synonymous with the term ‘coercion’?

(a) Implicit type casting.
(b) Explicit type casting.
(c) Implicit narrowing conversion.
(d) Implicit widening conversion.
(e) Explicit conversion.

Questions 46 – 50 refer to the following C program

```
// switchfun.c

#include <stdio.h>

int switchFun(int i)
{
   switch (i)
   {
      default: i += i;
      case  3: i += i;
      case  2: i += i;
      case  1: i += i;
      case  0: i  = i;
  }
  return i;

}

// Order of precedence (highest to lowest): {<<, >, ?:}.
// Recall that (a<<b) is ‘a’ left shifted ‘b’ bits.

int shiftFun(int i)
{
    return i << 4 > i ? i : 4;
}

int main(void)
{
   int i;

   for (i = 0; i < 10; i++)
      printf("%i => %i (%i)\n", i, switchFun(i), shiftFun(i));

   system("pause");
   return 0;
}
```

#46 What value is returned by **`SwitchFun(5)`**?

(a) 10.
(b) 80.
(c) 160.
(d) 384.
(e) None of the above.

#47 Which of the following would be returned by **`SwitchFun(1000)`**?

(a) 128
(b) 1000
(c) 32768
(d) 64000
(e) None of the above.

#48 As given, what will be returned by **`ShiftFun(1000)`**?

(a) 128
(b) 1000
(c) 32768
(d) 64000
(e) None of the above.

#49 What order of evaluation, first to last, must be followed by the **`ShiftFun()`** return expression in order to be equivalent to **`SwitchFun()`**?

(a) **`<<`**, **`?:`**, **`>`**.
(b) **`<<`**, **`>`**, **`?:`**.
(c) **`>`**, **`?:`**, **`<<`**.
(d) **`?:`**, **`>`**, **`<<`**.
(e) None of the above.

#50 Which of the following single-paren return expressions will make **`ShiftFun()`**equivalent to **`SwitchFun()`**?

(a) **`return (i<<4)>i?i:4;`**
(b) **`return (i<<4>i)?i:4;`**
(c) **`return i<<(4>i)?i:4;`**
(d) **`return i<<(4>i?i:4);`**
(e) **`return i<<4>(i?i:4);`**

Choose the BEST answer of those given and enter your choice on the Answer Sheet. You may choose multiple options, but the point value will be divided by the number of options chosen.

#1 In most C-Based languages, which is the following order of precedence (from highest to lowest):
(a) postfix increment/decrement; unary subtraction; multiplication; binary addition.
(b) multiplication/division; unary addition/subtraction; binary addition/subtraction.
(c) exponentiation; division; subtraction.
(d) binary subtraction; remainder; unary addition; exponentiation.
(e) multiplication; division; addition; subtraction.

#2 When two operators of the same precedence are adjacent in an expression, evaluation is performed
(a) left to right.
(b) right to left.
(c) according to the order of precedence of the operators.
(d) using the higher priority operator first.
(e) according to the associativity of the operators.

#3 What is a "side effect"?
(a) When the value of an expression must be delayed until another expression is evaluated.
(b) When the evaluation of an expression causes unintended consequences.
(c) When the order of evaluation of function's parameters affects the value of the expression.
(d) When the evaluation of an expression changes the value of a variable.
(e) When an expression cannot be fully evaluated until after all of its parameters are evaluated.

#4 Which of the following statements is NOT true regarding operator overloading:
(a) Operator overloading can harm the readability of a program.
(b) Operator overloading is resolved based on the types of the operands seen by the operator.
(c) Operator overloading can improve the readability of a program.
(d) Most languages inherently overload some operators.
(e) Overloaded operators always operate on user-defined data types.

#5 In C, if a=6, b=4, and c=2, the result of evaluating a < b < c will be
(a) -1
(b) True
(c) 1
(d) False
(e) 0

#6 An assignment statement with a conditional target means that

(a) the variable that a value is assigned to depends on the evaluation of an expression.

(b) the value of an expression may be assigned to multiple variables.

(c) the value that is assigned to a variable depends on the evaluation of an expression.

(d) the target may or may not be written to depending on the value of the expression to be written.

(e) the expression being evaluated depends on which variable the value will be assigned to.

#7 The ambiguous-else (or dangling-else) problem is generally resolved by what rule?

(a) each 'else' clause is paired with the most recent 'then' clause.

(b) each 'else' clause is paired with the most recent 'if' expression.

(c) each 'else' clause is paired with the most recent unpaired 'if' expression.

(d) each 'else' clause is matched to the most recent 'if' expression at the same level of indentation.

(e) each 'else' clause must be written so as to be unambiguous.

#8 The multiple selection statement is commonly known as

(a) the decision-based execution structure.

(b) the if-then-else structure.

(c) the controlled-execution structure.

(d) the if-goto structure.

(e) the switch structure.

#9 When control flows from one case label to the next, this is known as

(a) multi-case execution.

(b) a syntax error.

(c) a logic error.

(d) case ambiguity.

(e) flow-through.

#10 In C, if a switch expression does not have a 'default' case, then

(a) the first case segment is executed by default.

(b) the last case segment is executed by default.

(c) a compile-time error results.

(d) the entire construct is skipped if none of the case labels match the expression.

(e) a run-time exception is thrown.

#11 Iterative statements are broadly categorized as belong to one of which two categories?
(a) Internally-controlled and externally-controlled.
(b) Sentinel-controlled and Logically controlled.
(c) Counter-controlled and Logically controlled.
(d) Fixed-pattern and flexible-pattern.
(e) mutable and immutable.

#12 In C, which of the following is true of the looping constructs?
(a) All of looping constructs are required in order to have a complete language.
(b) Any loop that can be written using one construct can be implemented using either of the remaining constructs.
(c) The for loop can only be used for counter-controlled loops.
(d) The loop body can contain either 'break' or 'continue' statements, but never both.
(e) The do loop forces execution of the loop body and this cannot be accomplished using a while loop.

#13 With the C 'for' statement, which of the following statements is NOT true?
(a) All three of the expressions in the header are optional and could all three be left out.
(b) The 'for' statement is essentially a more formalized implementation of the 'while' statement.
(c) It is legal to branch into the middle of the loop body.
(d) The value of the loop variable can be changed within the body of the loop code.
(e) An infinite loop results if the 'for' header's the test expression (the middle expression) is left out.

#14 Most modern languages still include a 'goto' statement (or equivalent) for what reason?
(a) They don't -- modern languages almost always leave out the 'goto' statement entirely.
(b) It it not possible to implement a truly general purpose language without it.
(c) Eliminating it would break at least some previously written code.
(d) There are situations in which the judicious use of a 'goto' statement is reasonable and remains readable.
(e) The language designers usually choose to accommodate "traditional" programming styles.

#15 In addition to sequences, what two control structures are absolutely required to express computation?
(a) Pre-test loops and post-test loops.
(b) Selection and pre-test loops.
(c) Selection and post-test loops.
(d) Goto statements and selection statements.
(e) Selection statements and counter-controlled loops.

#16 Data-based control structures designed to execute the loop body one time for each data element are generally known as
(a) logic-controlled.
(b) data-controlled.
(c) collections.
(d) iterators.
(e) counter-controlled.

#17 In many processors, particularly simple ones, all of the control structures in a language are ultimately implemented using
(a) loop and branch instruction
(b) selection and iteration instructions.
(c) 'if' and 'while' instructions.
(d) stack-based branch instructions.
(e) conditional and unconditional branch (i.e., goto) instructions.

#18 Why do most languages include more control statements than are absolutely required for completeness?
(a) to eliminate ambiguity.
(b) backwards compatability.
(c) for simplicity.
(d) Readability and writability.
(e) consistency with common usage.

#19 The method used by some languages to eliminate the ambiguous-if problem syntactically is to require
(a) that any nested 'if' clause only follow unpaired 'else' clauses.
(b) that all selection statements have both a 'then' and an 'else' clause.
(c) the use of a reserved word to close the selection entity explicitly.
(d) that all 'else' clauses be compound statements.
(e) that any nested 'if' clause only follow paired 'else' clauses.

#20 Except for coroutines, subprograms in modern languages generally have all of the following characteristics except:
(a) control always returns to the caller when subprogram execution terminates.
(b) a single entry point.
(c) the calling program is suspended during execution of the called routine.
(d) only one subprogram is active at any given time.
(e) a single exit point.

#21 The general distinction between a "procedure" and a "function" is that
(a) procedures have access to all local variables of the caller.
(b) procedures do not take any arguments.
(c) functions return values while procedures do not.
(d) procedures can return multiple values while functions can only return a single (though possibly compound) value.
(e) functions are not allowed to have side effects.

#22 In object-oriented languages, a "method" generally refers to a subprogram that is
(a) associated with a class but need not be associated with an instance of that class.
(b) equivalent to a procedure in non objected-oriented languages.
(c) callable from anywhere in a program and is not associated with any particular class.
(d) equivalent to a function in non objected-oriented languages.
(e) called from an instance of class and has access to all of that instance's data.

#23 A generic subprogram is one whose
(a) calling environment includes all variables in all of its dynamic parents.
(b) computation can be done on data of different types in different calls.
(c) calling environment includes all variables in all of its static parents.
(d) computation solves a general problem instead of a specific problem.
(e) parameters are typeless.

#24 A 'closure' is a
(a) means of hiding information from the caller of a nested program.
(b) call to a nested subprogram that includes the necessary referencing environment.
(c) data structure that tracks the the referencing environment of one or more nested subprograms.
(d) nested subprogram combined with its referencing environment.
(e) set of subprograms that can all call each other.

#25 Which parameter passing scheme is also known as pass-by-copy?
(a) Pass-by-name
(b) Pass-by-result
(c) Pass-by-reference
(d) Pass-by-value-result
(e) Pass-by-value

#26 Which of the following parameter passing schemes gives gives rise to issues associated with aliases?
(a) pass-by-name
(b) pass-by-value
(c) pass-by-result
(d) pass-by-reference
(e) pass-by-value-result

#27 Because C uses row-major array indexing, which of the following is a workable header for passing an 20-row, 10-col array to a function?
(a) void fun(int matrix[10][20]) {...}
(b) void fun(int matrix[][10]) {...}
(c) void fun(int matrix[][20]) {...}
(d) void fun(int matrix[20][]) {...}
(e) void fun(int matrix[10][]) {...}

#28 The method of establishing the referencing environment for nested subprograms in statically-scoped languages is almost always
(a) shallow binding.
(b) lexical binding.
(c) referential binding.
(d) deep binding.
(e) ad hoc binding.

#29 An activation record is a data structure that holds all but which of the following information about a function call?
(a) Static local variables.
(b) Dynamic local variables.
(c) Parameters.
(d) Return addresses.
(e) Dynamic Links.

#30 In an activation record, the pointer to the base of the activation record of the caller is known as
(a) the return address.
(b) a static link.
(c) a dynamic link.
(d) the parent link.
(e) the child link.

#31 In an activation record, the pointer to the base of the activation record for the most recently called lexical parent of the called program is known as

(a) a static link.
(b) the return address.
(c) the parent link.
(d) a dynamic link.
(e) the child link.

#32 What is the only difference, in terms of the activation record instances, when a subprogram makes a recursive call to itself?

(a) The dynamic link will be self referential.
(b) There is no difference.
(c) The parent link will be self referential.
(d) The child link will be self referential.
(e) The static link will be self referential.

#33 The dynamic link, in static-scoped languages, is used to

(a) access the nonlocal referencing environment in the case of a closure.
(b) access nonlocal variables.
(c) provide traceback information when a run-time error occurs.
(d) return program control to the caller when the subprogram terminates.
(e) provide access to global variables.

#34 The dynamic link, in dynamic-scoped languages, is used to

(a) access nonlocal variables.
(b) provide traceback information when a run-time error occurs.
(c) access the nonlocal referencing environment in the case of a closure.
(d) return program control to the caller when the subprogram terminates.
(e) provide access to global variables.

#35 Which element is not needed in an activation record when subprograms cannot be nested?

(a) A dynamic link.
(b) Local variables.
(c) Parameters.
(d) The return address.
(e) A static link.

#36 Blocks (compound statements) that provide user-specified local scopes can be implemented as

(a) anonymous functions.

(b) parameterless nested subprograms, even in languages that don't support nested subprograms.

(c) auto-generated subprograms.

(d) closures.

(e) paramaterless nested subprograms, provided the language supports nested subprograms.

#37 Blocks within a subprogram can be implemented

(a) by including any block variables as part of the subprogram's set of local variables on the stack.

(b) as auto-generated subprograms.

(c) paramaterless nested subprograms, provided the language supports nested subprograms.

(d) as closures.

(e) as anonymous functions.

#38 Dynamic scoping that is implemented via tracing through the activation records for the nearest instance of a variable name is known as

(a) static access.

(b) dynamic access.

(c) deep access.

(d) stack access.

(e) shallow access.

#39 Dynamic scoping that is implemented via maintaining a separate stack for each named variable is known as

(a) shallow access.

(b) deep access.

(c) stack access.

(d) static access.

(e) dynamic access.

#40 A "central table" is one alternative method of implementing

(a) stack access.

(b) shallow access.

(c) static access.

(d) deep access.

(e) dynamic access.

#41 In some processors, particularly RISC processors, subprogram parameters are not passed on the stack but, instead, via

(a) static memory locations.
(b) linked lists maintained by the processor.
(c) a central table.
(d) processor registers.
(e) a separate parameter stack.

#42 A subprogram is 'active'

(a) from the time it is called until the time it terminates.
(b) whenever its static parent is active.
(c) only when its activation record is on top of the stack.
(d) from the time it is called until the main program terminates.
(e) only when its code is being executed.

#43 Which of the following operators is usually NOT right associative?

(a) The address dereference operator (in C: '*****').
(b) The pre-increment operator (in C: '**++**').
(c) The post-decrement operator (in C: '**--**').
(d) The bitwise negation (in C: '**~**').
(e) The unary minus operator (in C: '**-**').

#44 What value is output by the following C program?

```
#include <stdio.h>
int main(void)
{
    int a = 5, b = 10, c = -1, d;

    if ( (a-- < 10) && (b-- > 10) && (c++) )
       d = (a + 2*b)*c;
    else
       d = (a - 2*b)*c;

    printf("%i\n", d);
    return 0;
}
```

(a) -24.
(b) -22.
(c) 0.
(d) 14.
(e) 16.

#45 Which of the following is synonymous with the term 'coercion'?

(a) Implicit type casting.
(b) Explicit type casting.
(c) Implicit narrowing conversion.
(d) Implicit widening conversion.
(e) Explicit conversion.

Questions 46 – 50 refer to the following C program

```
// switchfun.c

#include <stdio.h>

int switchFun(int i)
{
   switch (i)
   {
      default: i += i;
      case  3: i += i;
      case  2: i += i;
      case  1: i += i;
      case  0: i  = i;
  }
  return i;

}

// Order of precedence (highest to lowest): {<<, >, ?:}.
// Recall that (a<<b) is 'a' left shifted 'b' bits.

int shiftFun(int i)
{
    return i << 4 > i ? i : 4;
}

int main(void)
{
   int i;

   for (i = 0; i < 10; i++)
      printf("%i => %i (%i)\n", i, switchFun(i), shiftFun(i));

   system("pause");
   return 0;
}
```

#46 What value is returned by **`SwitchFun(5)`**?

(a) 10.
(b) 80.
(c) 160.
(d) 384.
(e) None of the above.

#47 Which of the following would be returned by **`SwitchFun(1000)`**?

(a) 128
(b) 1000
(c) 32768
(d) 64000
(e) None of the above.

#48 As given, what will be returned by **`ShiftFun(1000)`**?

(a) 128
(b) 1000
(c) 32768
(d) 64000
(e) None of the above.

#49 What order of evaluation, first to last, must be followed by the **`ShiftFun()`** return expression in order to be equivalent to **`SwitchFun()`**?

(a) **`<<`**, **`?:`**, **`>`**.
(b) **`<<`**, **`>`**, **`?:`**.
(c) **`>`**, **`?:`**, **`<<`**.
(d) **`?:`**, **`>`**, **`<<`**.
(e) None of the above.

#50 Which of the following single-paren return expressions will make **`ShiftFun()`**equivalent to **`SwitchFun()`**?

(a) **`return (i<<4)>i?i:4;`**
(b) **`return (i<<4>i)?i:4;`**
(c) **`return i<<(4>i)?i:4;`**
(d) **`return i<<(4>i?i:4);`**
(e) **`return i<<4>(i?i:4);`**

Choose the BEST answer of those given and enter your choice on the Answer Sheet. You may choose multiple options, but the point value will be divided by the number of options chosen.

#1 In most C-Based languages, which is the following order of precedence (from highest to lowest):
(a) postfix increment/decrement; unary subtraction; multiplication; binary addition.
(b) multiplication/division; unary addition/subtraction; binary addition/subtraction.
(c) exponentiation; division; subtraction.
(d) binary subtraction; remainder; unary addition; exponentiation.
(e) multiplication; division; addition; subtraction.

#2 When two operators of the same precedence are adjacent in an expression, evaluation is performed
(a) left to right.
(b) right to left.
(c) according to the order of precedence of the operators.
(d) using the higher priority operator first.
(e) according to the associativity of the operators.

#3 What is a "side effect"?
(a) When the value of an expression must be delayed until another expression is evaluated.
(b) When the evaluation of an expression causes unintended consequences.
(c) When the order of evaluation of function's parameters affects the value of the expression.
(d) When the evaluation of an expression changes the value of a variable.
(e) When an expression cannot be fully evaluated until after all of its parameters are evaluated.

#4 Which of the following statements is NOT true regarding operator overloading:
(a) Operator overloading can harm the readability of a program.
(b) Operator overloading is resolved based on the types of the operands seen by the operator.
(c) Operator overloading can improve the readability of a program.
(d) Most languages inherently overload some operators.
(e) Overloaded operators always operate on user-defined data types.

#5 In C, if a=6, b=4, and c=2, the result of evaluating a < b < c will be
(a) -1
(b) True
(c) 1
(d) False
(e) 0

#6 An assignment statement with a conditional target means that

(a) the variable that a value is assigned to depends on the evaluation of an expression.

(b) the value of an expression may be assigned to multiple variables.

(c) the value that is assigned to a variable depends on the evaluation of an expression.

(d) the target may or may not be written to depending on the value of the expression to be written.

(e) the expression being evaluated depends on which variable the value will be assigned to.

#7 The ambiguous-else (or dangling-else) problem is generally resolved by what rule?

(a) each 'else' clause is paired with the most recent 'then' clause.

(b) each 'else' clause is paired with the most recent 'if' expression.

(c) each 'else' clause is paired with the most recent unpaired 'if' expression.

(d) each 'else' clause is matched to the most recent 'if' expression at the same level of indentation.

(e) each 'else' clause must be written so as to be unambiguous.

#8 The multiple selection statement is commonly known as

(a) the decision-based execution structure.

(b) the if-then-else structure.

(c) the controlled-execution structure.

(d) the if-goto structure.

(e) the switch structure.

#9 When control flows from one case label to the next, this is known as

(a) multi-case execution.

(b) a syntax error.

(c) a logic error.

(d) case ambiguity.

(e) flow-through.

#10 In C, if a switch expression does not have a 'default' case, then

(a) the first case segment is executed by default.

(b) the last case segment is executed by default.

(c) a compile-time error results.

(d) the entire construct is skipped if none of the case labels match the expression.

(e) a run-time exception is thrown.

#11 Iterative statements are broadly categorized as belong to one of which two categories?

(a) Internally-controlled and externally-controlled.

(b) Sentinel-controlled and Logically controlled.

(c) Counter-controlled and Logically controlled.

(d) Fixed-pattern and flexible-pattern.

(e) mutable and immutable.

#12 In C, which of the following is true of the looping constructs?

(a) All of looping constructs are required in order to have a complete language.

(b) Any loop that can be written using one construct can be implemented using either of the remaining constructs.

(c) The for loop can only be used for counter-controlled loops.

(d) The loop body can contain either 'break' or 'continue' statements, but never both.

(e) The do loop forces execution of the loop body and this cannot be accomplished using a while loop.

#13 With the C 'for' statement, which of the following statements is NOT true?

(a) All three of the expressions in the header are optional and could all three be left out.

(b) The 'for' statement is essentially a more formalized implementation of the 'while' statement.

(c) It is legal to branch into the middle of the loop body.

(d) The value of the loop variable can be changed within the body of the loop code.

(e) An infinite loop results if the 'for' header's the test expression (the middle expression) is left out.

#14 Most modern languages still include a 'goto' statement (or equivalent) for what reason?

(a) They don't -- modern languages almost always leave out the 'goto' statement entirely.

(b) It it not possible to implement a truly general purpose language without it.

(c) Eliminating it would break at least some previously written code.

(d) There are situations in which the judicious use of a 'goto' statement is reasonable and remains readable.

(e) The language designers usually choose to accommodate "traditional" programming styles.

#15 In addition to sequences, what two control structures are absolutely required to express computation?

(a) Pre-test loops and post-test loops.

(b) Selection and pre-test loops.

(c) Selection and post-test loops.

(d) Goto statements and selection statements.

(e) Selection statements and counter-controlled loops.

#16 Data-based control structures designed to execute the loop body one time for each data element are generally known as
(a) logic-controlled.
(b) data-controlled.
(c) collections.
(d) iterators.
(e) counter-controlled.

#17 In many processors, particularly simple ones, all of the control structures in a language are ultimately implemented using
(a) loop and branch instruction
(b) selection and iteration instructions.
(c) 'if' and 'while' instructions.
(d) stack-based branch instructions.
(e) conditional and unconditional branch (i.e., goto) instructions.

#18 Why do most languages include more control statements than are absolutely required for completeness?
(a) to eliminate ambiguity.
(b) backwards compatability.
(c) for simplicity.
(d) Readability and writability.
(e) consistency with common usage.

#19 The method used by some languages to eliminate the ambiguous-if problem syntactically is to require
(a) that any nested 'if' clause only follow unpaired 'else' clauses.
(b) that all selection statements have both a 'then' and an 'else' clause.
(c) the use of a reserved word to close the selection entity explicitly.
(d) that all 'else' clauses be compound statements.
(e) that any nested 'if' clause only follow paired 'else' clauses.

#20 Except for coroutines, subprograms in modern languages generally have all of the following characteristics except:
(a) control always returns to the caller when subprogram execution terminates.
(b) a single entry point.
(c) the calling program is suspended during execution of the called routine.
(d) only one subprogram is active at any given time.
(e) a single exit point.

#21 The general distinction between a "procedure" and a "function" is that
(a) procedures have access to all local variables of the caller.
(b) procedures do not take any arguments.
(c) functions return values while procedures do not.
(d) procedures can return multiple values while functions can only return a single (though possibly compound) value.
(e) functions are not allowed to have side effects.

#22 In object-oriented languages, a "method" generally refers to a subprogram that is
(a) associated with a class but need not be associated with an instance of that class.
(b) equivalent to a procedure in non objected-oriented languages.
(c) callable from anywhere in a program and is not associated with any particular class.
(d) equivalent to a function in non objected-oriented languages.
(e) called from an instance of class and has access to all of that instance's data.

#23 A generic subprogram is one whose
(a) calling environment includes all variables in all of its dynamic parents.
(b) computation can be done on data of different types in different calls.
(c) calling environment includes all variables in all of its static parents.
(d) computation solves a general problem instead of a specific problem.
(e) parameters are typeless.

#24 A 'closure' is a
(a) means of hiding information from the caller of a nested program.
(b) call to a nested subprogram that includes the necessary referencing environment.
(c) data structure that tracks the the referencing environment of one or more nested subprograms.
(d) nested subprogram combined with its referencing environment.
(e) set of subprograms that can all call each other.

#25 Which parameter passing scheme is also known as pass-by-copy?
(a) Pass-by-name
(b) Pass-by-result
(c) Pass-by-reference
(d) Pass-by-value-result
(e) Pass-by-value

#26 Which of the following parameter passing schemes gives gives rise to issues associated with aliases?
(a) pass-by-name
(b) pass-by-value
(c) pass-by-result
(d) pass-by-reference
(e) pass-by-value-result

#27 Because C uses row-major array indexing, which of the following is a workable header for passing an 20-row, 10-col array to a function?
(a) void fun(int matrix[10][20]) {...}
(b) void fun(int matrix[][10]) {...}
(c) void fun(int matrix[][20]) {...}
(d) void fun(int matrix[20][]) {...}
(e) void fun(int matrix[10][]) {...}

#28 The method of establishing the referencing environment for nested subprograms in statically-scoped languages is almost always
(a) shallow binding.
(b) lexical binding.
(c) referential binding.
(d) deep binding.
(e) ad hoc binding.

#29 An activation record is a data structure that holds all but which of the following information about a function call?
(a) Static local variables.
(b) Dynamic local variables.
(c) Parameters.
(d) Return addresses.
(e) Dynamic Links.

#30 In an activation record, the pointer to the base of the activation record of the caller is known as
(a) the return address.
(b) a static link.
(c) a dynamic link.
(d) the parent link.
(e) the child link.

#31 In an activation record, the pointer to the base of the activation record for the most recently called lexical parent of the called program is known as

(a) a static link.
(b) the return address.
(c) the parent link.
(d) a dynamic link.
(e) the child link.

#32 What is the only difference, in terms of the activation record instances, when a subprogram makes a recursive call to itself?

(a) The dynamic link will be self referential.
(b) There is no difference.
(c) The parent link will be self referential.
(d) The child link will be self referential.
(e) The static link will be self referential.

#33 The dynamic link, in static-scoped languages, is used to

(a) access the nonlocal referencing environment in the case of a closure.
(b) access nonlocal variables.
(c) provide traceback information when a run-time error occurs.
(d) return program control to the caller when the subprogram terminates.
(e) provide access to global variables.

#34 The dynamic link, in dynamic-scoped languages, is used to

(a) access nonlocal variables.
(b) provide traceback information when a run-time error occurs.
(c) access the nonlocal referencing environment in the case of a closure.
(d) return program control to the caller when the subprogram terminates.
(e) provide access to global variables.

#35 Which element is not needed in an activation record when subprograms cannot be nested?

(a) A dynamic link.
(b) Local variables.
(c) Parameters.
(d) The return address.
(e) A static link.

#36 Blocks (compound statements) that provide user-specified local scopes can be implemented as

(a) anonymous functions.

(b) parameterless nested subprograms, even in languages that don't support nested subprograms.

(c) auto-generated subprograms.

(d) closures.

(e) paramaterless nested subprograms, provided the language supports nested subprograms.

#37 Blocks within a subprogram can be implemented

(a) by including any block variables as part of the subprogram's set of local variables on the stack.

(b) as auto-generated subprograms.

(c) paramaterless nested subprograms, provided the language supports nested subprograms.

(d) as closures.

(e) as anonymous functions.

#38 Dynamic scoping that is implemented via tracing through the activation records for the nearest instance of a variable name is known as

(a) static access.

(b) dynamic access.

(c) deep access.

(d) stack access.

(e) shallow access.

#39 Dynamic scoping that is implemented via maintaining a separate stack for each named variable is known as

(a) shallow access.

(b) deep access.

(c) stack access.

(d) static access.

(e) dynamic access.

#40 A "central table" is one alternative method of implementing

(a) stack access.

(b) shallow access.

(c) static access.

(d) deep access.

(e) dynamic access.

#41 In some processors, particularly RISC processors, subprogram parameters are not passed on the stack but, instead, via

(a) static memory locations.
(b) linked lists maintained by the processor.
(c) a central table.
(d) processor registers.
(e) a separate parameter stack.

#42 A subprogram is 'active'

(a) from the time it is called until the time it terminates.
(b) whenever its static parent is active.
(c) only when its activation record is on top of the stack.
(d) from the time it is called until the main program terminates.
(e) only when its code is being executed.

#43 Which of the following operators is usually NOT right associative?

(a) The address dereference operator (in C: '*****').
(b) The pre-increment operator (in C: '**++**').
(c) The post-decrement operator (in C: '**--**').
(d) The bitwise negation (in C: '**~**').
(e) The unary minus operator (in C: '**-**').

#44 What value is output by the following C program?

```
#include <stdio.h>
int main(void)
{
    int a = 5, b = 10, c = -1, d;

    if ( (a-- < 10) && (b-- > 10) && (c++) )
       d = (a + 2*b)*c;
    else
       d = (a - 2*b)*c;

    printf("%i\n", d);
    return 0;
}
```

(a) -24.
(b) -22.
(c) 0.
(d) 14.
(e) 16.

#45 Which of the following is synonymous with the term ‘coercion’?

(a) Implicit type casting.
(b) Explicit type casting.
(c) Implicit narrowing conversion.
(d) Implicit widening conversion.
(e) Explicit conversion.

Questions 46 – 50 refer to the following C program

```
// switchfun.c

#include <stdio.h>

int switchFun(int i)
{
   switch (i)
   {
      default: i += i;
      case  3: i += i;
      case  2: i += i;
      case  1: i += i;
      case  0: i  = i;
  }
  return i;

}

// Order of precedence (highest to lowest): {<<, >, ?:}.
// Recall that (a<<b) is ‘a’ left shifted ‘b’ bits.

int shiftFun(int i)
{
    return i << 4 > i ? i : 4;
}

int main(void)
{
   int i;

   for (i = 0; i < 10; i++)
      printf("%i => %i (%i)\n", i, switchFun(i), shiftFun(i));

   system("pause");
   return 0;
}
```

#46 What value is returned by **`SwitchFun(5)`**?

(a) 10.
(b) 80.
(c) 160.
(d) 384.
(e) None of the above.

#47 Which of the following would be returned by **`SwitchFun(1000)`**?

(a) 128
(b) 1000
(c) 32768
(d) 64000
(e) None of the above.

#48 As given, what will be returned by **`ShiftFun(1000)`**?

(a) 128
(b) 1000
(c) 32768
(d) 64000
(e) None of the above.

#49 What order of evaluation, first to last, must be followed by the **`ShiftFun()`** return expression in order to be equivalent to **`SwitchFun()`**?

(a) **`<<`**, **`?:`**, **`>`**.
(b) **`<<`**, **`>`**, **`?:`**.
(c) **`>`**, **`?:`**, **`<<`**.
(d) **`?:`**, **`>`**, **`<<`**.
(e) None of the above.

#50 Which of the following single-paren return expressions will make **`ShiftFun()`**equivalent to **`SwitchFun()`**?

(a) **`return (i<<4)>i?i:4;`**
(b) **`return (i<<4>i)?i:4;`**
(c) **`return i<<(4>i)?i:4;`**
(d) **`return i<<(4>i?i:4);`**
(e) **`return i<<4>(i?i:4);`**

Choose the BEST answer of those given and enter your choice on the Answer Sheet. You may choose multiple options, but the point value will be divided by the number of options chosen.

#1 In most C-Based languages, which is the following order of precedence (from highest to lowest):

(a) postfix increment/decrement; unary subtraction; multiplication; binary addition.
(b) multiplication/division; unary addition/subtraction; binary addition/subtraction.
(c) exponentiation; division; subtraction.
(d) binary subtraction; remainder; unary addition; exponentiation.
(e) multiplication; division; addition; subtraction.

#2 When two operators of the same precedence are adjacent in an expression, evaluation is performed

(a) left to right.
(b) right to left.
(c) according to the order of precedence of the operators.
(d) using the higher priority operator first.
(e) according to the associativity of the operators.

#3 What is a "side effect"?

(a) When the value of an expression must be delayed until another expression is evaluated.
(b) When the evaluation of an expression causes unintended consequences.
(c) When the order of evaluation of function's parameters affects the value of the expression.
(d) When the evaluation of an expression changes the value of a variable.
(e) When an expression cannot be fully evaluated until after all of its parameters are evaluated.

#4 Which of the following statements is NOT true regarding operator overloading:

(a) Operator overloading can harm the readability of a program.
(b) Operator overloading is resolved based on the types of the operands seen by the operator.
(c) Operator overloading can improve the readability of a program.
(d) Most languages inherently overload some operators.
(e) Overloaded operators always operate on user-defined data types.

#5 In C, if a=6, b=4, and c=2, the result of evaluating a < b < c will be

(a) -1
(b) True
(c) 1
(d) False
(e) 0

#6 An assignment statement with a conditional target means that

(a) the variable that a value is assigned to depends on the evaluation of an expression.

(b) the value of an expression may be assigned to multiple variables.

(c) the value that is assigned to a variable depends on the evaluation of an expression.

(d) the target may or may not be written to depending on the value of the expression to be written.

(e) the expression being evaluated depends on which variable the value will be assigned to.

#7 The ambiguous-else (or dangling-else) problem is generally resolved by what rule?

(a) each 'else' clause is paired with the most recent 'then' clause.

(b) each 'else' clause is paired with the most recent 'if' expression.

(c) each 'else' clause is paired with the most recent unpaired 'if' expression.

(d) each 'else' clause is matched to the most recent 'if' expression at the same level of indentation.

(e) each 'else' clause must be written so as to be unambiguous.

#8 The multiple selection statement is commonly known as

(a) the decision-based execution structure.

(b) the if-then-else structure.

(c) the controlled-execution structure.

(d) the if-goto structure.

(e) the switch structure.

#9 When control flows from one case label to the next, this is known as

(a) multi-case execution.

(b) a syntax error.

(c) a logic error.

(d) case ambiguity.

(e) flow-through.

#10 In C, if a switch expression does not have a 'default' case, then

(a) the first case segment is executed by default.

(b) the last case segment is executed by default.

(c) a compile-time error results.

(d) the entire construct is skipped if none of the case labels match the expression.

(e) a run-time exception is thrown.

#11 Iterative statements are broadly categorized as belong to one of which two categories?
(a) Internally-controlled and externally-controlled.
(b) Sentinel-controlled and Logically controlled.
(c) Counter-controlled and Logically controlled.
(d) Fixed-pattern and flexible-pattern.
(e) mutable and immutable.

#12 In C, which of the following is true of the looping constructs?
(a) All of looping constructs are required in order to have a complete language.
(b) Any loop that can be written using one construct can be implemented using either of the remaining constructs.
(c) The for loop can only be used for counter-controlled loops.
(d) The loop body can contain either 'break' or 'continue' statements, but never both.
(e) The do loop forces execution of the loop body and this cannot be accomplished using a while loop.

#13 With the C 'for' statement, which of the following statements is NOT true?
(a) All three of the expressions in the header are optional and could all three be left out.
(b) The 'for' statement is essentially a more formalized implementation of the 'while' statement.
(c) It is legal to branch into the middle of the loop body.
(d) The value of the loop variable can be changed within the body of the loop code.
(e) An infinite loop results if the 'for' header's the test expression (the middle expression) is left out.

#14 Most modern languages still include a 'goto' statement (or equivalent) for what reason?
(a) They don't -- modern languages almost always leave out the 'goto' statement entirely.
(b) It it not possible to implement a truly general purpose language without it.
(c) Eliminating it would break at least some previously written code.
(d) There are situations in which the judicious use of a 'goto' statement is reasonable and remains readable.
(e) The language designers usually choose to accommodate "traditional" programming styles.

#15 In addition to sequences, what two control structures are absolutely required to express computation?
(a) Pre-test loops and post-test loops.
(b) Selection and pre-test loops.
(c) Selection and post-test loops.
(d) Goto statements and selection statements.
(e) Selection statements and counter-controlled loops.

#16 Data-based control structures designed to execute the loop body one time for each data element are generally known as
(a) logic-controlled.
(b) data-controlled.
(c) collections.
(d) iterators.
(e) counter-controlled.

#17 In many processors, particularly simple ones, all of the control structures in a language are ultimately implemented using
(a) loop and branch instruction
(b) selection and iteration instructions.
(c) 'if' and 'while' instructions.
(d) stack-based branch instructions.
(e) conditional and unconditional branch (i.e., goto) instructions.

#18 Why do most languages include more control statements than are absolutely required for completeness?
(a) to eliminate ambiguity.
(b) backwards compatability.
(c) for simplicity.
(d) Readability and writability.
(e) consistency with common usage.

#19 The method used by some languages to eliminate the ambiguous-if problem syntactically is to require
(a) that any nested 'if' clause only follow unpaired 'else' clauses.
(b) that all selection statements have both a 'then' and an 'else' clause.
(c) the use of a reserved word to close the selection entity explicitly.
(d) that all 'else' clauses be compound statements.
(e) that any nested 'if' clause only follow paired 'else' clauses.

#20 Except for coroutines, subprograms in modern languages generally have all of the following characteristics except:
(a) control always returns to the caller when subprogram execution terminates.
(b) a single entry point.
(c) the calling program is suspended during execution of the called routine.
(d) only one subprogram is active at any given time.
(e) a single exit point.

#21 The general distinction between a "procedure" and a "function" is that
(a) procedures have access to all local variables of the caller.
(b) procedures do not take any arguments.
(c) functions return values while procedures do not.
(d) procedures can return multiple values while functions can only return a single (though possibly compound) value.
(e) functions are not allowed to have side effects.

#22 In object-oriented languages, a "method" generally refers to a subprogram that is
(a) associated with a class but need not be associated with an instance of that class.
(b) equivalent to a procedure in non objected-oriented languages.
(c) callable from anywhere in a program and is not associated with any particular class.
(d) equivalent to a function in non objected-oriented languages.
(e) called from an instance of class and has access to all of that instance's data.

#23 A generic subprogram is one whose
(a) calling environment includes all variables in all of its dynamic parents.
(b) computation can be done on data of different types in different calls.
(c) calling environment includes all variables in all of its static parents.
(d) computation solves a general problem instead of a specific problem.
(e) parameters are typeless.

#24 A 'closure' is a
(a) means of hiding information from the caller of a nested program.
(b) call to a nested subprogram that includes the necessary referencing environment.
(c) data structure that tracks the the referencing environment of one or more nested subprograms.
(d) nested subprogram combined with its referencing environment.
(e) set of subprograms that can all call each other.

#25 Which parameter passing scheme is also known as pass-by-copy?
(a) Pass-by-name
(b) Pass-by-result
(c) Pass-by-reference
(d) Pass-by-value-result
(e) Pass-by-value

#26 Which of the following parameter passing schemes gives gives rise to issues associated with aliases?
(a) pass-by-name
(b) pass-by-value
(c) pass-by-result
(d) pass-by-reference
(e) pass-by-value-result

#27 Because C uses row-major array indexing, which of the following is a workable header for passing an 20-row, 10-col array to a function?
(a) void fun(int matrix[10][20]) {...}
(b) void fun(int matrix[][10]) {...}
(c) void fun(int matrix[][20]) {...}
(d) void fun(int matrix[20][]) {...}
(e) void fun(int matrix[10][]) {...}

#28 The method of establishing the referencing environment for nested subprograms in statically-scoped languages is almost always
(a) shallow binding.
(b) lexical binding.
(c) referential binding.
(d) deep binding.
(e) ad hoc binding.

#29 An activation record is a data structure that holds all but which of the following information about a function call?
(a) Static local variables.
(b) Dynamic local variables.
(c) Parameters.
(d) Return addresses.
(e) Dynamic Links.

#30 In an activation record, the pointer to the base of the activation record of the caller is known as
(a) the return address.
(b) a static link.
(c) a dynamic link.
(d) the parent link.
(e) the child link.

#31 In an activation record, the pointer to the base of the activation record for the most recently called lexical parent of the called program is known as

(a) a static link.
(b) the return address.
(c) the parent link.
(d) a dynamic link.
(e) the child link.

#32 What is the only difference, in terms of the activation record instances, when a subprogram makes a recursive call to itself?

(a) The dynamic link will be self referential.
(b) There is no difference.
(c) The parent link will be self referential.
(d) The child link will be self referential.
(e) The static link will be self referential.

#33 The dynamic link, in static-scoped languages, is used to

(a) access the nonlocal referencing environment in the case of a closure.
(b) access nonlocal variables.
(c) provide traceback information when a run-time error occurs.
(d) return program control to the caller when the subprogram terminates.
(e) provide access to global variables.

#34 The dynamic link, in dynamic-scoped languages, is used to

(a) access nonlocal variables.
(b) provide traceback information when a run-time error occurs.
(c) access the nonlocal referencing environment in the case of a closure.
(d) return program control to the caller when the subprogram terminates.
(e) provide access to global variables.

#35 Which element is not needed in an activation record when subprograms cannot be nested?

(a) A dynamic link.
(b) Local variables.
(c) Parameters.
(d) The return address.
(e) A static link.

#36 Blocks (compound statements) that provide user-specified local scopes can be implemented as

(a) anonymous functions.
(b) parameterless nested subprograms, even in languages that don't support nested subprograms.
(c) auto-generated subprograms.
(d) closures.
(e) paramaterless nested subprograms, provided the language supports nested subprograms.

#37 Blocks within a subprogram can be implemented

(a) by including any block variables as part of the subprogram's set of local variables on the stack.
(b) as auto-generated subprograms.
(c) paramaterless nested subprograms, provided the language supports nested subprograms.
(d) as closures.
(e) as anonymous functions.

#38 Dynamic scoping that is implemented via tracing through the activation records for the nearest instance of a variable name is known as

(a) static access.
(b) dynamic access.
(c) deep access.
(d) stack access.
(e) shallow access.

#39 Dynamic scoping that is implemented via maintaining a separate stack for each named variable is known as

(a) shallow access.
(b) deep access.
(c) stack access.
(d) static access.
(e) dynamic access.

#40 A "central table" is one alternative method of implementing

(a) stack access.
(b) shallow access.
(c) static access.
(d) deep access.
(e) dynamic access.

#41 In some processors, particularly RISC processors, subprogram parameters are not passed on the stack but, instead, via

(a) static memory locations.
(b) linked lists maintained by the processor.
(c) a central table.
(d) processor registers.
(e) a separate parameter stack.

#42 A subprogram is 'active'

(a) from the time it is called until the time it terminates.
(b) whenever its static parent is active.
(c) only when its activation record is on top of the stack.
(d) from the time it is called until the main program terminates.
(e) only when its code is being executed.

#43 Which of the following operators is usually NOT right associative?

(a) The address dereference operator (in C: '*****').
(b) The pre-increment operator (in C: '**++**').
(c) The post-decrement operator (in C: '**--**').
(d) The bitwise negation (in C: '**~**').
(e) The unary minus operator (in C: '**-**').

#44 What value is output by the following C program?

```
#include <stdio.h>
int main(void)
{
    int a = 5, b = 10, c = -1, d;

    if ( (a-- < 10) && (b-- > 10) && (c++) )
       d = (a + 2*b)*c;
    else
       d = (a - 2*b)*c;

    printf("%i\n", d);
    return 0;
}
```

(a) -24.
(b) -22.
(c) 0.
(d) 14.
(e) 16.

#45 Which of the following is synonymous with the term 'coercion'?

(a) Implicit type casting.
(b) Explicit type casting.
(c) Implicit narrowing conversion.
(d) Implicit widening conversion.
(e) Explicit conversion.

Questions 46 – 50 refer to the following C program

```
// switchfun.c

#include <stdio.h>

int switchFun(int i)
{
   switch (i)
   {
      default: i += i;
      case  3: i += i;
      case  2: i += i;
      case  1: i += i;
      case  0: i  = i;
  }
  return i;

}

// Order of precedence (highest to lowest): {<<, >, ?:}.
// Recall that (a<<b) is 'a' left shifted 'b' bits.

int shiftFun(int i)
{
    return i << 4 > i ? i : 4;
}

int main(void)
{
   int i;

   for (i = 0; i < 10; i++)
      printf("%i => %i (%i)\n", i, switchFun(i), shiftFun(i));

   system("pause");
   return 0;
}
```

#46 What value is returned by **`SwitchFun(5)`**?

(a) 10.
(b) 80.
(c) 160.
(d) 384.
(e) None of the above.

#47 Which of the following would be returned by **`SwitchFun(1000)`**?

(a) 128
(b) 1000
(c) 32768
(d) 64000
(e) None of the above.

#48 As given, what will be returned by **`ShiftFun(1000)`**?

(a) 128
(b) 1000
(c) 32768
(d) 64000
(e) None of the above.

#49 What order of evaluation, first to last, must be followed by the **`ShiftFun()`** return expression in order to be equivalent to **`SwitchFun()`**?

(a) **`<<`**, **`?:`**, **`>`**.
(b) **`<<`**, **`>`**, **`?:`**.
(c) **`>`**, **`?:`**, **`<<`**.
(d) **`?:`**, **`>`**, **`<<`**.
(e) None of the above.

#50 Which of the following single-paren return expressions will make **`ShiftFun()`**equivalent to **`SwitchFun()`**?

(a) **`return (i<<4)>i?i:4;`**
(b) **`return (i<<4>i)?i:4;`**
(c) **`return i<<(4>i)?i:4;`**
(d) **`return i<<(4>i?i:4);`**
(e) **`return i<<4>(i?i:4);`**

Choose the BEST answer of those given and enter your choice on the Answer Sheet. You may choose multiple options, but the point value will be divided by the number of options chosen.

#1 In most C-Based languages, which is the following order of precedence (from highest to lowest):
(a) postfix increment/decrement; unary subtraction; multiplication; binary addition.
(b) multiplication/division; unary addition/subtraction; binary addition/subtraction.
(c) exponentiation; division; subtraction.
(d) binary subtraction; remainder; unary addition; exponentiation.
(e) multiplication; division; addition; subtraction.

#2 When two operators of the same precedence are adjacent in an expression, evaluation is performed
(a) left to right.
(b) right to left.
(c) according to the order of precedence of the operators.
(d) using the higher priority operator first.
(e) according to the associativity of the operators.

#3 What is a "side effect"?
(a) When the value of an expression must be delayed until another expression is evaluated.
(b) When the evaluation of an expression causes unintended consequences.
(c) When the order of evaluation of function's parameters affects the value of the expression.
(d) When the evaluation of an expression changes the value of a variable.
(e) When an expression cannot be fully evaluated until after all of its parameters are evaluated.

#4 Which of the following statements is NOT true regarding operator overloading:
(a) Operator overloading can harm the readability of a program.
(b) Operator overloading is resolved based on the types of the operands seen by the operator.
(c) Operator overloading can improve the readability of a program.
(d) Most languages inherently overload some operators.
(e) Overloaded operators always operate on user-defined data types.

#5 In C, if a=6, b=4, and c=2, the result of evaluating a < b < c will be
(a) -1
(b) True
(c) 1
(d) False
(e) 0

#6 An assignment statement with a conditional target means that

(a) the variable that a value is assigned to depends on the evaluation of an expression.

(b) the value of an expression may be assigned to multiple variables.

(c) the value that is assigned to a variable depends on the evaluation of an expression.

(d) the target may or may not be written to depending on the value of the expression to be written.

(e) the expression being evaluated depends on which variable the value will be assigned to.

#7 The ambiguous-else (or dangling-else) problem is generally resolved by what rule?

(a) each 'else' clause is paired with the most recent 'then' clause.

(b) each 'else' clause is paired with the most recent 'if' expression.

(c) each 'else' clause is paired with the most recent unpaired 'if' expression.

(d) each 'else' clause is matched to the most recent 'if' expression at the same level of indentation.

(e) each 'else' clause must be written so as to be unambiguous.

#8 The multiple selection statement is commonly known as

(a) the decision-based execution structure.

(b) the if-then-else structure.

(c) the controlled-execution structure.

(d) the if-goto structure.

(e) the switch structure.

#9 When control flows from one case label to the next, this is known as

(a) multi-case execution.

(b) a syntax error.

(c) a logic error.

(d) case ambiguity.

(e) flow-through.

#10 In C, if a switch expression does not have a 'default' case, then

(a) the first case segment is executed by default.

(b) the last case segment is executed by default.

(c) a compile-time error results.

(d) the entire construct is skipped if none of the case labels match the expression.

(e) a run-time exception is thrown.

#11 Iterative statements are broadly categorized as belong to one of which two categories?
(a) Internally-controlled and externally-controlled.
(b) Sentinel-controlled and Logically controlled.
(c) Counter-controlled and Logically controlled.
(d) Fixed-pattern and flexible-pattern.
(e) mutable and immutable.

#12 In C, which of the following is true of the looping constructs?
(a) All of looping constructs are required in order to have a complete language.
(b) Any loop that can be written using one construct can be implemented using either of the remaining constructs.
(c) The for loop can only be used for counter-controlled loops.
(d) The loop body can contain either 'break' or 'continue' statements, but never both.
(e) The do loop forces execution of the loop body and this cannot be accomplished using a while loop.

#13 With the C 'for' statement, which of the following statements is NOT true?
(a) All three of the expressions in the header are optional and could all three be left out.
(b) The 'for' statement is essentially a more formalized implementation of the 'while' statement.
(c) It is legal to branch into the middle of the loop body.
(d) The value of the loop variable can be changed within the body of the loop code.
(e) An infinite loop results if the 'for' header's the test expression (the middle expression) is left out.

#14 Most modern languages still include a 'goto' statement (or equivalent) for what reason?
(a) They don't -- modern languages almost always leave out the 'goto' statement entirely.
(b) It it not possible to implement a truly general purpose language without it.
(c) Eliminating it would break at least some previously written code.
(d) There are situations in which the judicious use of a 'goto' statement is reasonable and remains readable.
(e) The language designers usually choose to accommodate "traditional" programming styles.

#15 In addition to sequences, what two control structures are absolutely required to express computation?
(a) Pre-test loops and post-test loops.
(b) Selection and pre-test loops.
(c) Selection and post-test loops.
(d) Goto statements and selection statements.
(e) Selection statements and counter-controlled loops.

#16 Data-based control structures designed to execute the loop body one time for each data element are generally known as
(a) logic-controlled.
(b) data-controlled.
(c) collections.
(d) iterators.
(e) counter-controlled.

#17 In many processors, particularly simple ones, all of the control structures in a language are ultimately implemented using
(a) loop and branch instruction
(b) selection and iteration instructions.
(c) 'if' and 'while' instructions.
(d) stack-based branch instructions.
(e) conditional and unconditional branch (i.e., goto) instructions.

#18 Why do most languages include more control statements than are absolutely required for completeness?
(a) to eliminate ambiguity.
(b) backwards compatability.
(c) for simplicity.
(d) Readability and writability.
(e) consistency with common usage.

#19 The method used by some languages to eliminate the ambiguous-if problem syntactically is to require
(a) that any nested 'if' clause only follow unpaired 'else' clauses.
(b) that all selection statements have both a 'then' and an 'else' clause.
(c) the use of a reserved word to close the selection entity explicitly.
(d) that all 'else' clauses be compound statements.
(e) that any nested 'if' clause only follow paired 'else' clauses.

#20 Except for coroutines, subprograms in modern languages generally have all of the following characteristics except:
(a) control always returns to the caller when subprogram execution terminates.
(b) a single entry point.
(c) the calling program is suspended during execution of the called routine.
(d) only one subprogram is active at any given time.
(e) a single exit point.

#21 The general distinction between a "procedure" and a "function" is that
(a) procedures have access to all local variables of the caller.
(b) procedures do not take any arguments.
(c) functions return values while procedures do not.
(d) procedures can return multiple values while functions can only return a single (though possibly compound) value.
(e) functions are not allowed to have side effects.

#22 In object-oriented languages, a "method" generally refers to a subprogram that is
(a) associated with a class but need not be associated with an instance of that class.
(b) equivalent to a procedure in non objected-oriented languages.
(c) callable from anywhere in a program and is not associated with any particular class.
(d) equivalent to a function in non objected-oriented languages.
(e) called from an instance of class and has access to all of that instance's data.

#23 A generic subprogram is one whose
(a) calling environment includes all variables in all of its dynamic parents.
(b) computation can be done on data of different types in different calls.
(c) calling environment includes all variables in all of its static parents.
(d) computation solves a general problem instead of a specific problem.
(e) parameters are typeless.

#24 A 'closure' is a
(a) means of hiding information from the caller of a nested program.
(b) call to a nested subprogram that includes the necessary referencing environment.
(c) data structure that tracks the the referencing environment of one or more nested subprograms.
(d) nested subprogram combined with its referencing environment.
(e) set of subprograms that can all call each other.

#25 Which parameter passing scheme is also known as pass-by-copy?
(a) Pass-by-name
(b) Pass-by-result
(c) Pass-by-reference
(d) Pass-by-value-result
(e) Pass-by-value

#26 Which of the following parameter passing schemes gives gives rise to issues associated with aliases?

(a) pass-by-name
(b) pass-by-value
(c) pass-by-result
(d) pass-by-reference
(e) pass-by-value-result

#27 Because C uses row-major array indexing, which of the following is a workable header for passing an 20-row, 10-col array to a function?

(a) void fun(int matrix[10][20]) {...}
(b) void fun(int matrix[][10]) {...}
(c) void fun(int matrix[][20]) {...}
(d) void fun(int matrix[20][]) {...}
(e) void fun(int matrix[10][]) {...}

#28 The method of establishing the referencing environment for nested subprograms in statically-scoped languages is almost always

(a) shallow binding.
(b) lexical binding.
(c) referential binding.
(d) deep binding.
(e) ad hoc binding.

#29 An activation record is a data structure that holds all but which of the following information about a function call?

(a) Static local variables.
(b) Dynamic local variables.
(c) Parameters.
(d) Return addresses.
(e) Dynamic Links.

#30 In an activation record, the pointer to the base of the activation record of the caller is known as

(a) the return address.
(b) a static link.
(c) a dynamic link.
(d) the parent link.
(e) the child link.

#31 In an activation record, the pointer to the base of the activation record for the most recently called lexical parent of the called program is known as
(a) a static link.
(b) the return address.
(c) the parent link.
(d) a dynamic link.
(e) the child link.

#32 What is the only difference, in terms of the activation record instances, when a subprogram makes a recursive call to itself?
(a) The dynamic link will be self referential.
(b) There is no difference.
(c) The parent link will be self referential.
(d) The child link will be self referential.
(e) The static link will be self referential.

#33 The dynamic link, in static-scoped languages, is used to
(a) access the nonlocal referencing environment in the case of a closure.
(b) access nonlocal variables.
(c) provide traceback information when a run-time error occurs.
(d) return program control to the caller when the subprogram terminates.
(e) provide access to global variables.

#34 The dynamic link, in dynamic-scoped languages, is used to
(a) access nonlocal variables.
(b) provide traceback information when a run-time error occurs.
(c) access the nonlocal referencing environment in the case of a closure.
(d) return program control to the caller when the subprogram terminates.
(e) provide access to global variables.

#35 Which element is not needed in an activation record when subprograms cannot be nested?
(a) A dynamic link.
(b) Local variables.
(c) Parameters.
(d) The return address.
(e) A static link.

#36 Blocks (compound statements) that provide user-specified local scopes can be implemented as
(a) anonymous functions.
(b) parameterless nested subprograms, even in languages that don't support nested subprograms.
(c) auto-generated subprograms.
(d) closures.
(e) paramaterless nested subprograms, provided the language supports nested subprograms.

#37 Blocks within a subprogram can be implemented
(a) by including any block variables as part of the subprogram's set of local variables on the stack.
(b) as auto-generated subprograms.
(c) paramaterless nested subprograms, provided the language supports nested subprograms.
(d) as closures.
(e) as anonymous functions.

#38 Dynamic scoping that is implemented via tracing through the activation records for the nearest instance of a variable name is known as
(a) static access.
(b) dynamic access.
(c) deep access.
(d) stack access.
(e) shallow access.

#39 Dynamic scoping that is implemented via maintaining a separate stack for each named variable is known as
(a) shallow access.
(b) deep access.
(c) stack access.
(d) static access.
(e) dynamic access.

#40 A "central table" is one alternative method of implementing
(a) stack access.
(b) shallow access.
(c) static access.
(d) deep access.
(e) dynamic access.

#41 In some processors, particularly RISC processors, subprogram parameters are not passed on the stack but, instead, via

(a) static memory locations.
(b) linked lists maintained by the processor.
(c) a central table.
(d) processor registers.
(e) a separate parameter stack.

#42 A subprogram is 'active'

(a) from the time it is called until the time it terminates.
(b) whenever its static parent is active.
(c) only when its activation record is on top of the stack.
(d) from the time it is called until the main program terminates.
(e) only when its code is being executed.

#43 Which of the following operators is usually NOT right associative?

(a) The address dereference operator (in C: '*****').
(b) The pre-increment operator (in C: '**++**').
(c) The post-decrement operator (in C: '**--**').
(d) The bitwise negation (in C: '**~**').
(e) The unary minus operator (in C: '**-**').

#44 What value is output by the following C program?

```
#include <stdio.h>
int main(void)
{
    int a = 5, b = 10, c = -1, d;

    if ( (a-- < 10) && (b-- > 10) && (c++) )
       d = (a + 2*b)*c;
    else
       d = (a - 2*b)*c;

    printf("%i\n", d);
    return 0;
}
```

(a) -24.
(b) -22.
(c) 0.
(d) 14.
(e) 16.

#45 Which of the following is synonymous with the term ‘coercion’?

(a) Implicit type casting.
(b) Explicit type casting.
(c) Implicit narrowing conversion.
(d) Implicit widening conversion.
(e) Explicit conversion.

Questions 46 – 50 refer to the following C program

```
// switchfun.c

#include <stdio.h>

int switchFun(int i)
{
   switch (i)
   {
      default: i += i;
      case  3: i += i;
      case  2: i += i;
      case  1: i += i;
      case  0: i  = i;
  }
  return i;

}

// Order of precedence (highest to lowest): {<<, >, ?:}.
// Recall that (a<<b) is ‘a’ left shifted ‘b’ bits.

int shiftFun(int i)
{
    return i << 4 > i ? i : 4;
}

int main(void)
{
   int i;

   for (i = 0; i < 10; i++)
      printf("%i => %i (%i)\n", i, switchFun(i), shiftFun(i));

   system("pause");
   return 0;
}
```

#46 What value is returned by **`SwitchFun(5)`**?

(a) 10.
(b) 80.
(c) 160.
(d) 384.
(e) None of the above.

#47 Which of the following would be returned by **`SwitchFun(1000)`**?

(a) 128
(b) 1000
(c) 32768
(d) 64000
(e) None of the above.

#48 As given, what will be returned by **`ShiftFun(1000)`**?

(a) 128
(b) 1000
(c) 32768
(d) 64000
(e) None of the above.

#49 What order of evaluation, first to last, must be followed by the **`ShiftFun()`** return expression in order to be equivalent to **`SwitchFun()`**?

(a) **`<<`**, **`?:`**, **`>`**.
(b) **`<<`**, **`>`**, **`?:`**.
(c) **`>`**, **`?:`**, **`<<`**.
(d) **`?:`**, **`>`**, **`<<`**.
(e) None of the above.

#50 Which of the following single-paren return expressions will make **`ShiftFun()`**equivalent to **`SwitchFun()`**?

(a) **`return (i<<4)>i?i:4;`**
(b) **`return (i<<4>i)?i:4;`**
(c) **`return i<<(4>i)?i:4;`**
(d) **`return i<<(4>i?i:4);`**
(e) **`return i<<4>(i?i:4);`**