HW02 Problem Set

CS-3160

**Assignment**

HW02: Sebesta Problems 5.(4-12), Programming Exercise 5.7
HANDWRITTEN – Due at beginning of class on due date.
PROGRAM – Due at midnight, via Blackboard, at midnight on day prior to due date.

**Handwritten Portion**

4. Dynamic type binding is closely related to implicit heap-dynamic variables. Explain this relationship.
5. Describe a situation when a history-sensitive variable in a subprogram is useful.
6. Consider the following JavaScript skeletal program:

```
// The main program
var x;
function sub1() {
  var x;
  function sub2() {
    ...
  }
}
function sub3() {
  ...
}
```

Assume that the execution of this program is in the following unit order:

`main` calls `sub1`

`sub1` calls `sub2`

`sub2` calls `sub3`

a. Assuming static scoping, in the following, which declaration of x is the correct one for a reference to x?

   i. `sub1`
   ii. `sub2`
   iii. `sub3`

b. Repeat part a, but assume dynamic scoping.

7. Assume the following JavaScript program was interpreted using static-scoping rules. What value of x is displayed in function sub1? Under dynamic-scoping rules, what value of x is displayed in function sub1?

```
var x;
function sub1() {
  document.write("x = " + x + "<br />");
}
function sub2() {
  var x;
  x = 10;
  sub1();
}
x = 5;
sub2();
```

8. Consider the following JavaScript program:

```
var x, y, z;
function sub1() {
  var a, y, z;
  function sub2() {
    var a, b, z;
    ...
  }
  ...
}
function sub3() {
  var a, x, w;
  ...
}
```

List all the variables, along with the program units where they are declared, that are visible in the bodies of sub1, sub2, and sub3, assuming static scoping is used.

9. Consider the following Python program:

```
x = 1;
y = 3;
z = 5;
def sub1():
  a = 7;
  y = 9;
  z = 11;
  ...
def sub2():
  global x;
  a = 13;
  x = 15;
  w = 17;
  ...
  def sub3():
    nonlocal a;
    a = 19;
    b = 21;
    z = 23;
    ...
...
```

List all the variables, along with the program units where they are declared, that are visible in the bodies of `sub1`, `sub2`, and `sub3`, assuming static scoping is used.

10. Consider the following C program:

```
void fun(void) {
  int a, b, c; /* definition 1 */
  ...
  while (...) {
    int b, c, d; /*definition 2 */
    ... <---------------- 1
    while (...) {
     int c, d, e; /* definition 3 */
     ... <---------------- 2
     }
    ... <---------------- 3
    }
  ... <---------------- 4
}
```

For each of the four marked points in this function, list each visible variable, along with the number of the definition statement that defines it.

11. Consider the following skeletal C program:

```
void fun1(void);  /* prototype */
void fun2(void);  /* prototype */
void fun3(void);  /* prototype */
void main() {
  int a, b, c;
  ...
 }
void fun1(void) {
  int b, c, d;
  ...
 }
void fun2(void) {
  int c, d, e;
  ...
 }
void fun3(void) {
  int d, e, f;
  ...
 }
```

Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last function called? Include with each visible variable the name of the function in which it was defined.

a. `main` calls `fun1`; `fun1` calls `fun2`; `fun2` calls `fun3`.

b. `main` calls `fun1`; `fun1` calls `fun3`.

c. `main` calls `fun2`; `fun2` calls `fun3`; `fun3` calls `fun1`.

d. `main` calls `fun3`; `fun3` calls `fun1`.

e. `main` calls `fun1`; `fun1` calls `fun3`; `fun3` calls `fun2`.

f. `main` calls `fun3`; `fun3` calls `fun2`; `fun2` calls `fun1`.

12. Consider the following program, written in JavaScript-like syntax:

```
// main program
var x, y, z;

function sub1() {
  var a, y, z;
  ...
}
function sub2() {
  var a, b, z;
  ...
}
function sub3() {
  var a, x, w;
  ...
}
```

Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last subprogram activated? Include with each visible variable the name of the unit where it is declared.

a. `main` calls `sub1`; `sub1` calls `sub2`; `sub2` calls `sub3`.

b. `main` calls `sub1`; `sub1` calls `sub3`.

c. `main` calls `sub2`; `sub2` calls `sub3`; `sub3` calls `sub1`.

d. `main` calls `sub3`; `sub3` calls `sub1`.

e. `main` calls `sub1`; `sub1` calls `sub3`; `sub3` calls `sub2`.

f. `main` calls `sub3`; `sub3` calls `sub2`; `sub2` calls `sub1`.

**Programming Portion**

7. Write three functions in C or C++: one that declares a large array statically, one that declares the same large array on the stack, and one that creates the same large array from the heap. Call each of the subprograms a large number of times (at least 100,000) and output the time required by each. Explain the results.

Your main() function should call each of the three functions and determine the average execution time for each function. The function should not to anything except allocate the necessary memory resources – do not initialize the memory, the goal is to examine the impact of how the memory is bound to storage on the performance.

Your program should output the necessary information to the console and you should capture this information and past it into a ReadMe.txt file to be submitted along with your code files. Also in the ReadMe.txt should be your explanation of the results. There is no need for any graphs or figures, just a text explanation.

Your C program should be ANSI-C and/or C99 compliant – if you just use the standard library functions you should be okay. Each file should contain a header according to the header template linked on the Course Homepage (on GetTeched).

Place your source code (*.c and *.h) files and the ReadMe.txt file in a directory named HW02 and zip this directory up into a file names HW02.zip and submit electronically to Blackboard. It is fine to include any other files that your programming environment needs/creates (including the executable file) as long as the ZIP file does not exceed 1MB in size. In other words, you don't need to clear out a bunch of files or copy the submittal files someplace else unless you want to.

**Grading Rubric**

The assignment is worth 25 pts (as a whole) and the score will be recorded as a percentage of that amount.

| Problem | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|
| Points | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 |

Programming Problem: 5pts

10% Physical Format
50% Answers correct (and supported by work)
40% Effort evidenced by the submitted work