

**Due: July 6**

- \* ABSOLUTELY NO LATE ASSIGNMENTS!
- \* Homework solutions MUST be TYPED, except for diagrams, which may be hand-drawn.
- \* Limit your answers to at MOST half a page per question (10 or 12 pt font).
- Short, concise answers are best.
- \* Answer the questions IN YOUR OWN WORDS!

Total: 50 points

From **Problem Set** for Chapter 5 of the class textbook

#5. (2 points) Describe a situation when a history-sensitive variable in a subprogram is useful.

#8. (4 points) 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;
  . . .
}
```

Please 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.

#11. (6 points) 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 the 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. (6 points) 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 the 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.

From **Problem Set** for Chapter 6 of the class textbook

#2. (2 points) How does a decimal value waste memory space?

#15. (2 points) What are the arguments for and against Java's implicit heap storage recovery when compared with explicit heap storage recovery required in C++? Consider real-time systems.

From **Problem Set** for Chapter 7 of the class textbook

#8. (2 points) Describe a situation where the add operator in a programming language would not be commutative.

#13. (4 points) Let the function fun be defined as

```
int fun(int *k) {  
    *k += 4;  
    return 3 * (*k) - 1;  
}
```

Suppose fun is used in a program as follows:

```
void main(){  
    int i = 10, j = 10, sum1, sum2;  
    sum1 = (i/2) + fun(&i);  
    sum2 = fun(&j) + (j/2);  
}
```

What are the values of sum1 and sum2

- a. if the operands in the expressions are evaluated left to the right?
- b. if the operands in the expressions are evaluated right to the left?

\*\*\*\*\* Programming Homework (12 points) \*\*\*\*\*

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 subprogram many times (at least 100,000) and output the time required. Explain the results.

\*\*\*\*\* Programming Homework (12 points) \*\*\*\*\*

Write a C or C++ program that does a large number of references to elements of two-dimensional arrays, using only subscripts. Write a second program that does the same operation but uses pointers and pointer arithmetic to do the array references for the storage-mapping function. Compare the time efficiency of the two programs. Which of the two programs is likely to be more reliable? Why?