CSci 365: Organizations of Programming Languages

Name: _____

Assignment 4 (chap. 5 & 6): 130 points

Q1. [10] Assume the following JavaScript program was interpreted using static-scoping rules.

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

1) What value of x is displayed in function sub1?
2) Under dynamic-scoping rules, what value of x is displayed in function sub1?

Q2. [20] 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.

Point	Variables	Definition
e.g.) Point 1	a b c d	1 2 3 4
Point 1		
Point 2		

Point 3		
Point 4		

Q3. [20] 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;
. . .
}
```

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.

	variables	Unit where declared
In sub1:		
In sub2:		
In sub3:		

Q4. [20] 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.

1) `main` calls `sub1; sub1` calls `sub2; sub2` calls `sub3.`

variables	unit

2) `main` calls `sub2; sub2` calls `sub3; sub3` calls `sub1.`

variables	unit

Q5. [20] Multidimensional arrays can be stored in row-major order, as in C++, or in column-major order, as in Fortran. Develop the access functions for both of these arrangements for three-dimensional arrays.

Let the subscript ranges of the three dimensions be named [min(1), max(1)], [min(2), max(2)], and [min(3), max(3)]. Let the sizes of the subscript ranges be size(1), size(2), and size(3). Assume the element size is 1.

The base address of an array P, `base(P) = address of P[min(1), min(2), min(3)].`

1) Row Major Order: location(`P[i, j, k]`) = ?
2) Column Major Order: location(`P[i, j, k]`) = ?

Q6. [10] In the following C++ program, write the output values. Suppose the integer variable v1 is bound to the address of '0xbfc601ac' in the memory location.

NOTE: ' count << "Value of v1 : " << v1 << endl;' is a print statement such as 'print("Value of v1: ", v1, \n)'

```
int main () {
      int v1 = 20;
      int *ip;

      ip = &var;
```

```
    cout << "Value of v1 variable: " << v1 << endl;

    cout << "Address stored in ip variable: " << ip << endl;

    cout << "Value of *ip variable: " << *ip << endl;

    return 0;
}
```

Q7. [15] In the following C++ program, write the output values.

```
int main () {
    int     v2;
    double d;

    int     &r = v2;
    doubld &s = d;

    v2 = 5;
    cout << "Value of v2: " << v2 << endl;
    cout << "Value of v2 reference: " << r << endl;

    d = 11.7;
    cout << "Value of d: " << d << endl;
    cout << "Value of d reference: " << s << endl;

    return 0;
}
```

Q8. [15] In the following C++ program, write the output values.

```
void test(int*, int*);

int main () {
    int     a = 5, b = 5;

    cout << "Before calling test function: " << endl;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;

    test(&a, &b);

    cout << "\nAfter calling test function: " << endl;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;

    return 0;
```

```
}

void test(int *nl, int *n2) {
      *n1 = 10;
      *n2 = 11;
}
```