CISC 2000 — Computer Science II
Fall, 2018

Homework Assignment #2 (revised)

1 Recall the following functions and operators:

- sizeof function: returns the size of a variable (i.e., the number of bytes used by a variable).
  *N*ote when you **cout** a pointer variable or value, it will be shown in hexadecimal, as indicated by the **0X** prefix of the number. You can cast the pointer to **long** if you want the address to be displayed in decimal.

Write a C++ program that declares three variables and three pointers that pointing to them (as below), and add cout statements to display out the size, and address of the variables. Run your program and write down your answers in the blank.

```
int a=10;         // the size of a is: _________
double d=3.1415; // the size of d is: _________
char c='A';         // the size of c is: __________

char str[21]="Seeing is believing!"; // the size of str is: _________, the length of str is: ______
int numbers[10]={100, 101, 102};    // the size of numbers is: _________

int* p = NULL;  // the size of p is: _________.
   // the value of p is: ______________

double* q=NULL; // the size of q is:  __________
// the value of q is: _____________

char* t=NULL;  // the size of t is: ________, the value of t is: ___________

//Use cout statement to find out values of p, q, t
p = &a;    // the value of p is __________________
q = &d;   // the value of q is ________________
t = &c;  // the value of t is ________________

cout << *p << endl; // This displays ______ (be specific). When accessing *p, the
                // ___ bytes are accessed, and their addresses
// are from ___________________
// to ___________________.




cout << *q << endl; // This displays ______. When accessing *q,
                 // ___ bytes are accessed, and their addresses
 // are from ____________________
 // to ________________.



cout << *t << endl; // This displays ______. When accessing *t,
                 // ___ bytes are, and there addresses are from
 // are from ____________________
 // to ________________.



// Array vs poiner
cout << "numbers is" << numbers << endl;  //This displays: ____________________________
p = numbers;
cout << "p=" << p <<endl;    //This displays: _________________________________
int* p2 = &(numbers[0]);
cout << "p2=" <<p2 <<endl;   // This displays: _____________________
cout << "p+1=" << p+1 << endl;  // How much is p differs from p+1? ______
// p+1 is the address of which element in numbers:______
// p+2 is the address of which element in numbers: _________


cout << "q+1=" << q+1 << endl; // How much is q differs from q+1? ______

cout << "t+1=" << t+1 << endl; // How much is t differs from t+1? ______

// We observe that all pointer variables are of the same size, but we need to differentiate
// a pointer to an int against a pointer to a double, because:
___________________________________________________________________.




// char array is special, supported by library functions such as strlen(), strcpy(),
// input and output using cin/cout
t = str;  // What happens here? ________________
cout << t <<endl;  //What will be the output? Why?
//Hint: recall pointer is same as array
// cout <<t<<endl; is similar to

// char t[SIZE];
// cout << t <<endl;  //this will display the char array, C string

cout <<t+1 <<endl;  //What will be the output?
// Why?
```

2 Draw memory diagram to represent variables used in the following program, and trace the program to find out the output of the program.

```
#include <iostream>
using namespace std;


int main()
{
    int c=20;
    int a=10;
    int b[20]={1,2,3,4,5};

    int * d = new int;

    int * p = NULL;
   ///////////////////////////////////////////////
   // Draw all varilables to the right side of the program


   p = &a;

   *p = c;

   *d = *p+20;
   /////Redraw the current status of variables at this point

   cout <<"*d"<<endl;
   //what's the output of the statement above?


   p = d;
   *p = *p+100;
   cout <<"*d"<<endl;
   /////Redraw the current status of variables at this point
   //what's the output of the statement above?


   p = b;
   *p = 0;

   p++;
   *p = 0;
   /////Redraw the current status of variables at this point
   cout <<b[0]<<endl<<b[1]<<endl;
   //what's the output of the statement above?

}
```

3 Array and pointer exercises. Answer the following questions, and verify them by typing up, compiling and running the program.

```
int sum (int b[], int size)
{
   int total = 0;

   for (int i=0;i<size;i++)
    total+=b[i];

   return total;
}
```

```
int main()
{
   int a[5]={100,200,300,400,500};
   int *p= a;


   //Todo: Write a statement to use p to set a[0] to 1000



   //What's the difference of the two function calls if any? Explain and confirm your answer:
    int total1 = sum (a, 5);
    int total2 = sum (p, 5);







    //Todo: Write a statement to set p to points to a[2]



    //After the above statement (that sets p to poitns to a[2], What's the output of
    // the following statement?
    //Explain by tracing the function call
    cout <<sum (p,2) <<endl;



}
```

4 Finish the following program, test, and debug the code.

```
/* A function that reads a sequence of integers from input (with the length of sequence,
   followed by the numbers), and saves the numbers in an array of the given length
   For example, if the length is 3, and the numbers are
     123 345 99
   then the array returned will be of size 3, and stores values 123, 345, and 99
  @param length: upon return, stores the length/size of the array
  @return the pointer pointing to the array
*/
int * ReadNumberSequence (int & size)
{

   do {
     cout <<"Enter the length of the number sequence:";
     cin >> size;
   } while (size<=0);
   int array[size];



   // Todo: write a loop to read size # of int from input, and save them to the array




   return array;
}

int main()
{
// Todo: delcare necesssary variables



//Todo: call the ReadNumberSequence function to read a sequence of numbers




//Todo: write a loop to display the elements in the array returned ...






}
```

5 Draw memory diagram to illustrate all variables used in the following program.

(a) Structure/class type object and pointers

```
class Node{
   int a;
   Node* next;
};

int main()
{
    Node* p = NULL;
    Node* q = NULL;
    p = new Node;


    //Todo: What's the sizeof *p?

    (*p).a = 0;  //accessing member variables of the Node variable that p is pointing to
    (*p).next = NULL;

    q = new Node;
    q->a = 1;   //a shorthand for the combination of * and . above
    q->next = NULL;
    //Todo: Draw memory diagram to illustrate all variables at this point


    //store address of second node in the first node
    p->next = q;


    //Todo: Draw memory diagram to illustrate all variables at this point




    //Todo: allocate another node variable (dynamically)
    // and stores its address in the second node's next field







}
```

(b) Tracing a program and find out what it prints in the terminal.

```
int main()
{
    char stars[11]="**********";
    char spaces[11]="          "; //10 spaces

    for (int i=0;i<10;i++)
     cout << stars+i <<endl;

    for (int i=0;i<10;i++)
    {
         cout <<spaces+i<< stars+(10-i)<<endl;
    }
}
```

(c) array of arrays

```
typedef int* IntPtr;

int main()
{
    int a[3]={1,2,3};
    int b[2]={4,5};

    IntPtr arrayofPtr[2];

    arrayofPtr[0] = a;
    arrayofPtr[1] = b;

    // arrayOfPtr is now storing a "2D" arrays: first row is a, second row is b

    //Draw memory diagram at this point


    //Todo: use arrayofPtr to access b[1]



    // Todo: dynamically allocate two arrays of size 10, and stores
    // their addresses in arrayofPtr



    // arrayOfPtr is now storing a 2x10 "2D" arrays, an array of pointers
    // Write a loop to fill the array of arrays with 0





   //Draw the memory diagram at this point



}
```