Lab 02,

Pointers and Dynamic Arrays

OBJECTIVE:

Things that will be covered in today's lab:

- Pointers
- Dynamic Arrays

THEORY:

Pointer variable: A variable whose content is an address (i.e., a memory address).In C++, you declare a pointer variable by using the asterisk symbol (*) between the datatype and the variable name. The general syntax to declare a pointer variable is as follows:

```
datatype * identifier;
```

In C++, the ampersand (&), address of the operator, is a unary operator that returns the address of its operand. Similarly "*" is a dereferencing operator, refers to the object to which its operand (pointer) points. For example, given the statements:

```
int x;

int *p;

p = &x;      // assigns address of x to p

cout << *p << endl;  // pointer p points to x
```

The arrays discussed in last lab are called static arrays because their size was fixed at compile time. One of the limitations of a static array is that every time you execute the program, the size of the array is fixed. One way to handle this limitation is to declare an array that is large enough to process a variety of data sets. However, if the array is very big and the data set is small, such a declaration would result in memory waste. On the other hand, it would be helpful if, during program execution, you could prompt the user to enter the size of the array and then create an array of the appropriate size.

Dynamic Array: An array created during the execution of a program. To create a dynamic array, we use new operator.

```
int size;
```

int *p;

p = newint [size];

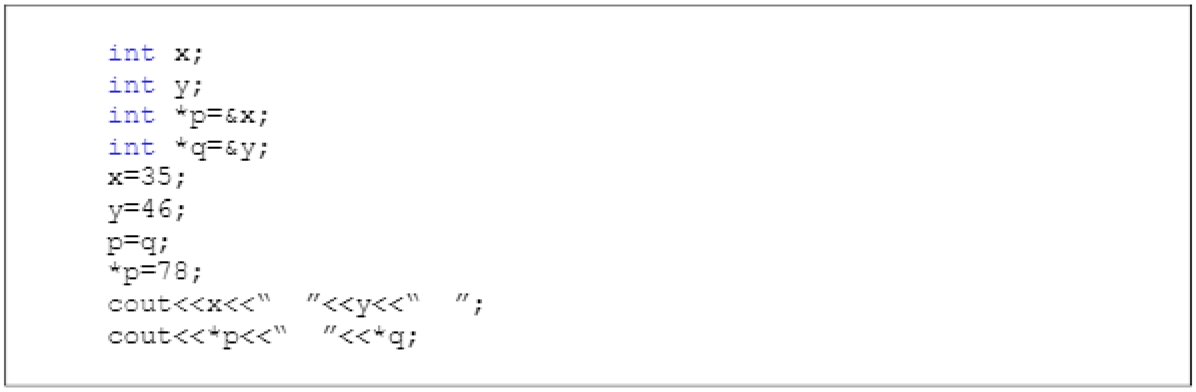
If you are not in need of dynamically allocated memory anymore, you can use delete operator, which de-allocates memory previously allocated by new operator.

delete [] p ;

Exercise 1:

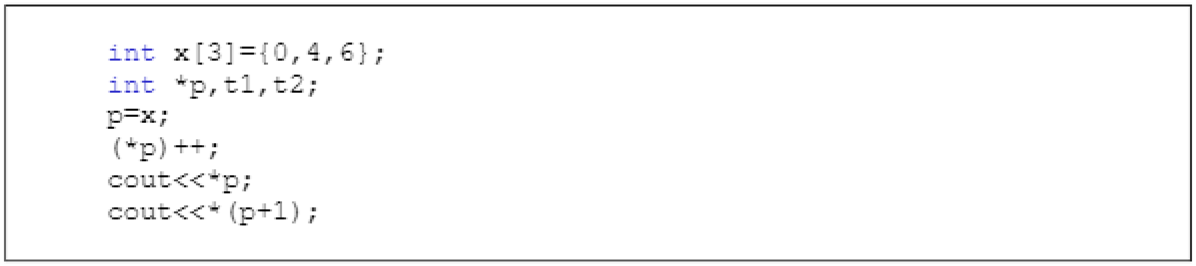
Write the output of the following C++ codes without running the code in Visual Studio

a)



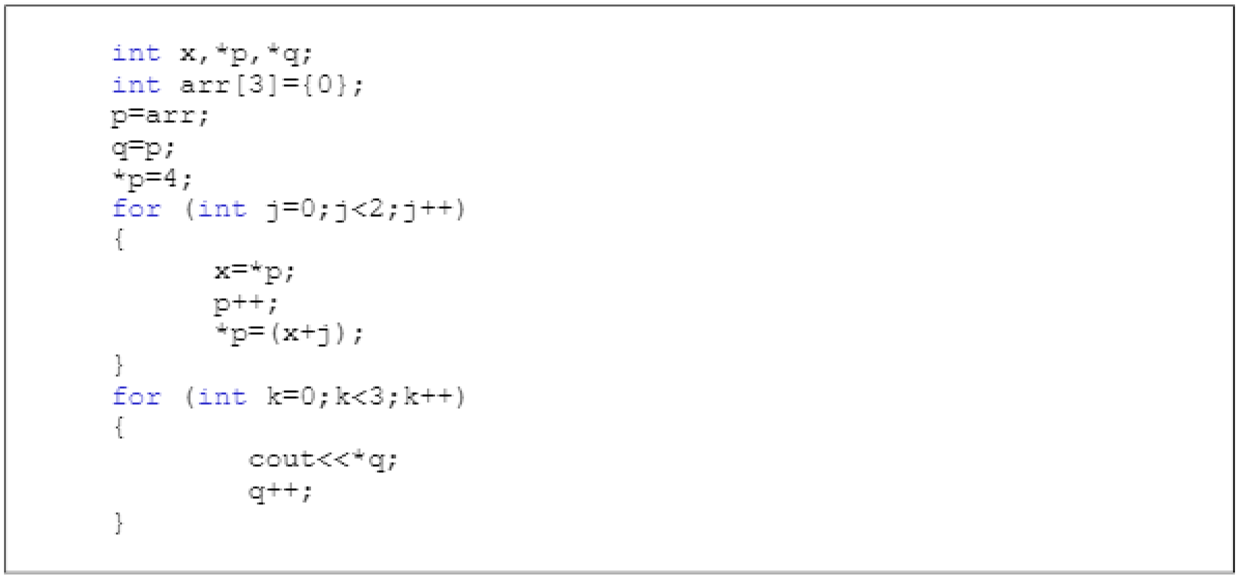
```
int x;
int y;
int *p=&x;
int *q=&y;
x=35;
y=46;
p=q;
*p=78;
cout<<x<<"  "<<y<<"  ";
cout<<*p<<"  "<<*q;
```

b)



```
int x[3]={0,4,6};
int *p,t1,t2;
p=x;
(*p)++;
cout<<*p;
cout<<*(p+1);
```

c)

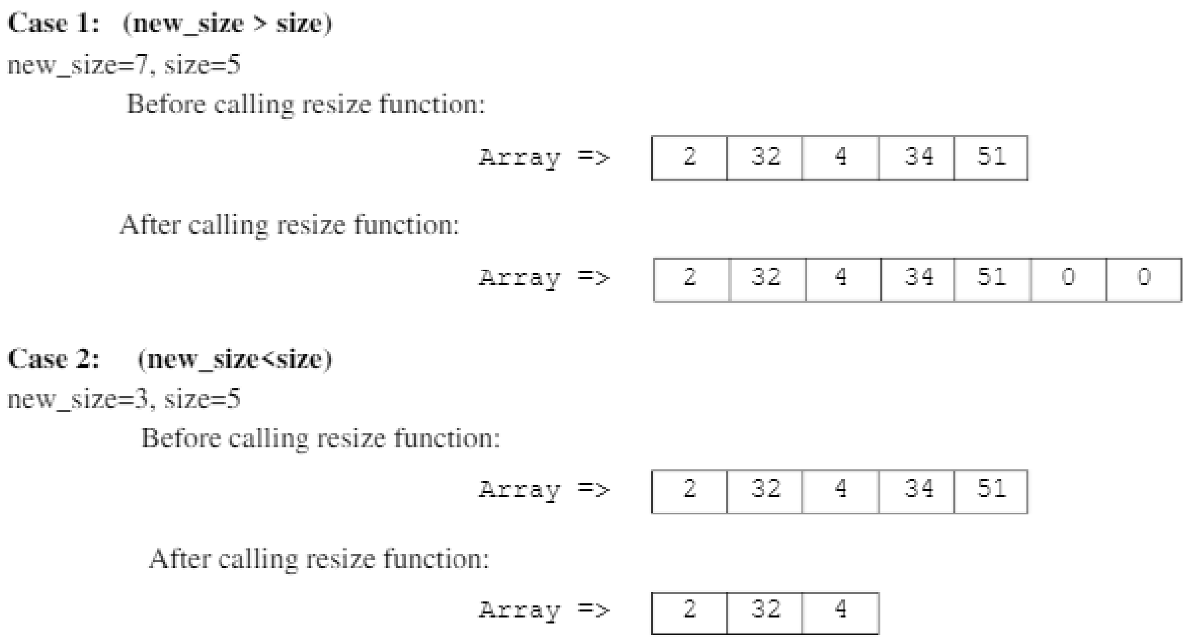


```
int x,*p,*q;
int arr[3]={0};
p=arr;
q=p;
*p=4;
for (int j=0;j<2;j++)
{
        x=*p;
        p++;
        *p=(x+j);
}
for (int k=0;k<3;k++)
{
           cout<<*q;
           q++;
}
```

Exercise 2:

Write a function resize() that takes as arguments: a pointer pointing to the array of integers, itssize, and a new_size. New_size can be any number greater than 0. This function should change the size of the array. If the new size is greater than the previous one, then insert zeroes in new cells**.**

Example:



Case 1: (new_size > size)

new_size=7, size=5

Before calling resize function:

Array =>

2	32	4	34	51

After calling resize function:

Array =>

2	32	4	34	51	0	0

Case 2: (new_size<size)

new_size=3, size=5

Before calling resize function:

Array =>

2	32	4	34	51

After calling resize function:

Array =>

2	32	4

Exercise 3:

Write a code that merges two arrays. Create two dynamic arrays of sizes *size_1* and *size_2* respectively. Take input in these arrays from the user. Now create a third array of *size* (*size_1+size_2*) and insert all the elements of both arrays in this array. Remove the duplicate elements from this array and resize the array to a smaller size.

Example:

Array 1=>	1	2	3	4	
Array 2=>	3	4	5	6	7

After merging Array1 and Array2:

Array 3=>	1	2	3	3	4	4	5	6	7

After removing duplicate elements, this array should be of size 6:

Array 3=>	1	2	3	4	5	6	7

Post Lab:

Consider following main function:

```
void main()
{
char input[100];
cin.getline(input,100);
//For example, user enters National University.

      char *q=input;
      ReverseSentence(q);
      cout<<input<<endl;

// Now input array should be changed to lanoitaNytisrevinU.
}
```

Write the implementation of the function **void ReverseSentence(char*)**. Assume that each sentence ends with a full stop. You should use the following function **ReverseWord()** to reverse each word of the whole sentence. You are not allowed to use any static array. You are only allowed to use simple character pointers.

```
void ReverseWord(char *p, intlen)
{
      char temp;
      for(int i=0; i<len/2; i++)
      {
            temp=p[i];
            p[i]=p[len-i-1];
            p[len-i-1]=temp;
            }
}
```

"p" is a pointer pointing to the first location of *char* array and length is the number of characters in the array. For example, if p is pointing to "HELLO" then the length is 5. After calling this function, p is pointing to "OLLEH".