Overview

This homework has a written part and a programming part. Questions in the written part revolve around pointers and dynamic memory. You will be able to answer all of them after Friday’s class. The programming part involves implementation of two addition member functions for the *SetOfInt* class.

The written part is due at the start of class on Tuesday, Sept. 25th. No late written homework will be accepted. It will be returned in class on Friday, Sept. 28th. The programming part is due Thursday evening, September 27th, at 11:59:59pm. See the syllabus for a discussion of late penalties on the programming part. You are to submit only the files *SetOfInt.h* and *SetOfInt.cpp*.

Please note that you must put your name and section number on every paper you turn in and in every file you submit electronically.

Written Questions

These written questions stress pointer manipulation and pointer arithmetic.

1. **(10 points)** What is the output from the following program? If you want to compile and run this (after typing it in), you are allowed to do so. Please try to answer it correctly by hand, first. (Remember: you will not have use of a computer on a test.)
#include <iostream>
using namespace std;

int
main()
{
    int *p;
    int x = 5;
p = &x;
    *p = 12;
cout << "A: x = " << x << " , *p = " << *p << " \n";

    int y = 18;
    int *q = &y;
p = q;
    *p = -4;
cout << "B: x = " << x << " , y = " << y << " \n";

    q = new int[ 4 ];
for ( int i=0; i<4; ++i ) q[i] = 3*i;
p = &(q[2]);
    *p = 33;
    *(p+1) = 66;
    *(q+1) = 55;
cout << "C: q = {";
for ( p = q; p != q+4; ++ p )
    cout << " " << *p;
    cout << " }\n";

return 0;
}

Solution:

A: x = 12, *p = 12
B: x = 12, y = -4
C: q = { 0 55 33 66 }

2. (10 pts) Write a function that is passed an array and then steps
through the array using pointer arithmetic to find the minimum and maximum values in the array. The function declaration is

```c
void max_and_min( float * values, int n,
                 float& max_value, float& min_value );
```

Solution:

```c
void max_and_min( float * values, int n,
                 float& max_value, float& min_value )
{
    max_value = *values;
    min_value = *values;
    for ( float * p = values+1; p<values+n; ++p )
    {
        if ( *p < min_value )
            min_value = *p;
        if ( *p > max_value )
            max_value = *p;
    }
}
```

3. (10 pts) In class we looked at a method of dynamically allocating a two dimensional array. It was something like:

```c
int nrows, ncols;
cout << "Enter the number of rows and columns => ";
cin >> nrows >> ncols;
float ** matrix;
matrix = new float* [ nrows ];
for ( int r=0; r<nrows; ++r )
    matrix[ r ] = new float[ ncols ];
```

Write a short segment of code to delete `matrix`, including all dynamically allocated memory.

Solution:

```c
for ( int r=0; r<nrows; ++r )
    delete [] matrix[ r ];
delete [] matrix;
```
4. (10 pts) What is the output of the following program? Why didn’t I delete the array pointed to by q at the end? Please note that this is a challenge question, harder than what will appear on a test. It pushes your understanding of pointers, arrays, and parameter passing.

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

void mystery( int *a, int* & b )
{
    *b = 5;
    *(a+2) = 7;
    int* temp = b;
    b = a;
    a = temp;
    b[1] = 6;
    a[0] = 3;
    a[1] = -5;
    int i;
    cout << "\nInside mystery:\n";
    cout << "a’s contents = {";
    for ( i=0; i<3; ++i ) cout << " " << a[i];
    cout << " }\n" << "b’s contents = {";
    for ( i=0; i<3; ++i ) cout << " " << b[i];
    cout << " }\n";
}

int main()
{
    int * p = new int[ 3 ];
    int * q = new int[ 3 ];
    int i;
    for ( i=0; i<3; ++i )
    {
        p[ i ] = i;
        q[ i ] = 10+i;
    }
}


cout << "Before mystery\n";
cout << "p’s contents = {";
for ( i=0; i<3; ++i ) cout << " " << p[i];
cout << " }\n" << "q’s contents = {";
for ( i=0; i<3; ++i ) cout << " " << q[i];
cout << " }\n";

mystery( p, q );

cout << "\nAfter mystery\n";
cout << "p’s contents = {";
for ( i=0; i<3; ++i ) cout << " " << p[i];
cout << " }\n" << "q’s contents = {";
for ( i=0; i<3; ++i ) cout << " " << q[i];
cout << " }\n";

delete [] p;

}  

Solution: Here is the output:
Programming Problems

Implement and test two additional member functions in the SetOfInt class:

- Write the set_union friend function. The union of two sets contains the elements that are in either set. For example, if

$$S = \{1, 56, 13, 22\} \quad \text{and} \quad T = \{79, 89, 56, -14, 22, 77\},$$

then the union is

$$S \cup T = \{1, 56, 13, 22, 79, 89, -14, 77\}.$$  

Elements that appear in both sets are not repeated.

- Write a new constructor that creates a set from an array of integers. The integers in this array are not necessarily unique. In other words, some integers could be in the array more than once. On the other hand, in the array pointed to by the elements member variable, no integer should appear more than once. The constructor should have the declaration

$$\text{SetOfInt( const int array_of_int[], int n );}$$

where \(n\) is the number of entries in the array.

Each of the functions is worth 15 points. See the Week 2 homework for general grading policies. Of the points for each function, 5 will be for compilation, 5 will be for structure, and 5 will be for correct execution. We will supply our own new main program, and you will NOT have the opportunity to see this before submitting your program.