CS 410 – Introduction to Software Engineering – Spring 2019
Instructor: Marc Pomplun

Midterm Practice Exam

Sample Solutions

No books, no notes, and no calculators are allowed.

Question 1: _____ out of ____ points
Question 2: _____ out of ____ points
Question 3: _____ out of ____ points
Question 4: _____ out of ____ points (bonus)

Total Score: / 58 points

Grade:
### Question 1: Some Warm-Up Questions

Tell whether each of the following statements is true or false by checking the appropriate box. Give it your best guess if you do not know the answer.

<table>
<thead>
<tr>
<th>Statement</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Every C++ program is object-oriented.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
<tr>
<td>b) Any C program can be successfully compiled by a C++ compiler without making any changes to the code.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
<tr>
<td>c) C++ classes can be nested, i.e., one class can be defined within another class.</td>
<td>[X]</td>
<td>[ ]</td>
</tr>
<tr>
<td>d) C++ allows operator overloading.</td>
<td>[X]</td>
<td>[ ]</td>
</tr>
<tr>
<td>e) In C++, every function is a member of a class.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
<tr>
<td>f) In C++, classes containing pure virtual functions cannot be instantiated.</td>
<td>[X]</td>
<td>[ ]</td>
</tr>
<tr>
<td>g) The C++ template mechanism allows to specify only one type per class.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
<tr>
<td>h) If a C++ class definition does not include a constructor, the compiler will automatically provide a default constructor.</td>
<td>[X]</td>
<td>[ ]</td>
</tr>
<tr>
<td>i) C++ programs require a main() function, which cannot be a member of any class.</td>
<td>[X]</td>
<td>[ ]</td>
</tr>
<tr>
<td>j) Every C++ class has to define a destructor.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
</tbody>
</table>

2 points for every correct answer
Question 2: A Mysterious Program

When the Visual Attention Lab at UMB moved to its new place, the lab team discovered an ancient C++ program written on an old, torn sheet of paper (see next page). Can you help the lab team to find out what each of the following functions/variables of the program does and what the program’s output looks like? Use English words for your descriptions; you can also draw a schematic sketch of the output if you cannot describe it (but do not draw the entire output!).

Mystery::char a[size][size]: (3 points)

This 50×50 character array serves as a canvas for the program to draw on. Initially it is blank (filled with dots), then the program draws Xs on it and finally prints the array on the screen.

Mystery::int x, y: (3 points)

(x, y) indicates the current position of the drawing cursor on the canvas. It is updated whenever an X is drawn.

Mystery::Mystery(int x0, int y0): (4 points)

This is the constructor for the Mystery class. It initializes the canvas by completely filling it with dots. Moreover, it sets the initial cursor position to (x0, y0).

Mystery::Draw(direction d, int length): (4 points)

Starting at position (x, y), this function draws a straight line of Xs on the canvas. Direction d specifies the direction (up, down, left, or right) and int length specifies the maximum length of the line. The line drawing also terminates if the cursor leaves the canvas. In that case, the function returns false, otherwise it returns true.

Mystery::GenerateOutput(): (4 points)

This function draws a sequence of lines in the cyclic order of directions left, up, right, and down. The first line is of length one, and each following line is one X longer than the preceding one. The function terminates as soon as the drawing cursor leaves the canvas.
What does the output look like? (6 points)

It looks like a spiral of Xs starting in the center of the canvas, as shown below, but extending to size 50×50:

............
.XXXXXXXXX.
.X       X.
.X     .X.X.
.X   .X.X.
.X.X .X.X.
.X  .X.X.
.X.X  X.X.
.X.X    X.
.X.XX    X.
.X.XX    X.
.X.XX    X.
.X.XX    X.
.X.XX    X.
.X.XX    X.
.X.XX    X.
.X.XX    X.
.X.XX    X.
.X.XX    X.
.X.XX    X.
.X.XX    X.
.X.XX    X.
#include <iostream>
using namespace std;

const int size = 50;

enum direction {UP, DOWN, LEFT, RIGHT};

class Mystery {
public:
    Mystery(int x0, int y0);
    void GenerateOutput();
private:
    bool Draw(direction d, int length);
    char a[size][size];
    int x, y;
};

Mystery::Mystery(int x0, int y0)
{
    x = x0;
    y = y0;
    for (int j = 0; j < size; j++)
        for (int i = 0; i < size; i++)
            a[i][j] = '.';
}

bool Mystery::Draw(direction d, int length)
{
    for (int i = 0; i < length; i++)
    {
        switch (d)
            {
        case UP:    a[x][y--] = 'X'; break;
        case DOWN:  a[x][y++] = 'X'; break;
        case LEFT:  a[x--][y] = 'X'; break;
        case RIGHT: a[x++][y] = 'X';
        }
        if (x < 0 || x >= size || y < 0 || y >= size)
            return false;
    }
    return true;
}

void Mystery::GenerateOutput()
{
    int length = 1;
    while (Draw(LEFT, length++) && Draw(UP, length++) &&
            Draw(RIGHT, length++) && Draw(DOWN, length++));
    for (int j = 0; j < size; j++)
    {
        for (int i = 0; i < size; i++)
            cout << a[i][j];
        cout << endl;
    }
}

int main()
{
    Mystery m(size/2, size/2);
    m.GenerateOutput();
}
Question 3: Chase the Bugs!

For his latest research project, Professor P. wants to write a program that sorts an integer array containing ten elements. Going for the simplest solution, he decides to implement a bubble sort algorithm. The basic idea of bubble sort is that you first compare the first two elements in the list (elements 1 and 2). If element 1 is greater than element 2, you swap the values of the two elements, that is, element 1 receives the value of element 2 and vice versa. If element 1 is less than or equal to element 2, nothing happens. You then do the same thing with elements 2 and 3, then elements 3 and 4, and so on, until elements 9 and 10. This way the greatest element in the list will be carried to the end of the list. You repeat the whole procedure, but this time you can stop after comparing elements 8 and 9, because you know that element 10 is in the desired position already. Next time you can stop after elements 7 and 8, and so on, until the whole list is sorted in ascending order.

As usual, Professor P. is in a hurry and seriously confused, so the first version of his program contains a lot of bugs. Please help him correct all the bugs so that the program creates the following desired output:

-3000
-54
0
1
16
22
34
512
1432
13245

Please indicate and correct all bugs in the program listing on the next page. Try to use as few changes as possible to make the program compile and run correctly. You may also insert and delete lines if you like to. Do not rewrite entire lines of code but try to keep the changes as small as possible.

(Of course this task is difficult to do without using a compiler. Try to find as many bugs as possible; you will receive bonus points if you actually manage to correct all the bugs.)
```cpp
#include <iostream>      // insert: #include <vector>
using namespace std;

void SwapIntegers(int &a, int &b)
{
    int temp = a;       // &a &b
    a = b;
    temp = b;     b = temp;
}

void BubbleSort(vector<int> &intVector)
{
    for (int i = intVector.size() - 1; i > 0; i--)   i--
        for (int j = 0; j < i; j++)      >
            if (intVector[j] < intVector[j + 1])
                SwapIntegers(intVector[j], intVector[j + 1]);
}

int main()
{
    int intArray[] = {34, 1432, 1, -54, 16, 22, 13245, 512, -3000, 0};
    vector<int> intVector(intArray, intArray + 9);
    BubbleSort(&intVector);    10
    
    for (int i = 0; i <= intVector.size(); i++)
        cout >> intVector[i] >> endl;
    return 0;
}
```

2 points for every identified bug, and -1 point for every newly introduced bug.
Question 4 (Bonus Question): Deep Learning

The idea of Artificial Neural Networks has been around for a long time, but Deep Learning is relatively new and has led to many successes. What are the main reasons that Deep Learning algorithms are possible to be trained today?

There are three main reasons that should be mentioned here:

- Vastly increased computing power through hardware acceleration (GPUs), large clusters, cloud computation, etc.,
- Advances in training algorithms (backpropagation), especially with regard to training very deep architectures (new activation functions, convolutional networks, dropout, etc.), and
- Availability of a variety huge datasets (millions of exemplars) necessary to train deep networks.

Maximum of 12 points total.