Lab 04 - Nodes

Instructions:

- **The lab requires completing a few tasks.**
- **Your submissions must be submitted to the Lab04 directory of your GitHub repository or uploaded to the Lab04 assignment on Google classroom.**
- **Accompanying these instructions are a few header files that must be included in the appropriate programs you have to write.**
- **Besides the header files provided, your programs can only include the libraries *iostream*, *string*, *fstream*, *sstream*, and *cctype*.**
- **Cheating of any kind is prohibited and will not be tolerated.**
- **Violating and/or failing to follow any of the rules will result in an automatic zero (0) for the lab.**

TO ACKNOWLEDGE THAT YOU HAVE READ AND UNDERSTOOD THE INSTRUCTIONS ABOVE, AT THE BEGINNING OF YOUR SUBMISSION(S), ADD A COMMENT THAT CONSISTS OF YOUR NAME AND THE DATE

Grading:

Task	Maximum Points	Points Earned
1	2	
2	3	
3	5	
Total	10	

Note: solutions will be provided for tasks colored blue only.

Task 1

In the file `"main.cpp"`, state what the function `F()` does as a comment above its definition; and then, define the function `G()` that is equivalent to `F()` but has an int *Array* reference parameter instead of an int *Node* pointer parameter.

Task 2

Create a file named `"table.xls"` that construct the runtime table of the function `Monotonic()` whose definition is below

```
bool Monotonic(const Array<int>& data)
{
  for(int i = 1;i < data.Size();i += 1)
  {
    if(data[i-1] > data[i])
    {
      return false;
    }
  }
  return true;
}
```

afterwards, in the file `"main.cpp"`, write the definition of the bool function `Monotonic()` whose header is

```
bool Monotonic(const Node<int>* root)
```

It should operate exactly like the `Monotonic()` array function above. Note that linked lists can contain no elements unlike arrays.

Task 3

Copy the Project.h file from Lab 03 to Lab 04, and modify the file as follows

- **define a class named *Game* that contains**
  - **a private string field named *board*.**
  - **a private bool array field named *states* with a size of 36.**
  - **a public default constructor that assigns a string consisting of 36 uppercase `'A'` characters to *board* and assigns false to each element of *states*.**
  - **a public copy constructor.**
  - **a public assignment operator.**
  - **a public empty destructor.**
  - **a public bool method named `ValidCoordinates()` that takes four int parameters named *rw1*, *cn1*, *rw2* and *cn2* respectively. It returns true only if all parameters are between 0 and 5 inclusively, and the values of the elements of *states* whose indices are equal to the corresponding two-dimensional indices represented by the pairs of parameters [(*rw1*, *cn1*) and (*rw2*, *cn2*)] are both false. The formula for converting a two-dimensional index to a one-dimensional index for *board* is**

    $$i = 6 \times x + y$$

    where i is the one-dimensional index and (x, y) are the two-dimensional index with x corresponding to a row parameter [*rw1* or *rw1*] and y corresponding to a column parameter [*cn1* or *cn2*].
- **move `BoardView()` into the class *Game* as a private method and remove its parameters. The function must work with the fields of the class.**