Due July 18th at midnight to eCampus

First Name McKenzie Last Name Burch UIN 225005240

User Name mburch13gig-em E-mail address mburch13gig-em@gmail.com

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more: Aggie Honor System Office

Type of sources			
People	Sara Wild	Grant Ballard (PT)	
Web pages (provide URL)	https://www.tutorialspoint.com http://www.cplusplus.com		
Printed material	Textbook		
Other Sources	Lecture Notes		

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.
"On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work."

Your Name McKenzie Burch Date July 18, 2018

Programming Assignment 2 (140 points)

1. (30 pts) Implement a stack ADT using STL vector to support the operations: `push()`, `pop()`, `top()`, `empty()`.
   - push(): add a new element on to the top of the stack
   - pop(): remove the element on the top of the stack
   - top(): returns the element on the top of the stack
   - empty(): returns a true boolean if the stack is empty
2. (10 pts) Test the operations of the stack class for correctness. What is the running time of each operation? Express it using the Big-O asymptotic notation.
   - push(): $f(n) = O(n)$
   - pop(): $f(n) = O(1)$
   - top(): $f(n) = O(1)$
   - empty(): $f(n) = O(1)$
3. Stack Application

   Use the implemented stack class as an auxiliary data structure to compute spans used in the financial analysis, e.g. to get the number of consecutive days when stack was growing.

Definition of the span:

Given a vector X, the span $S[i]$ of $X[i]$ is the maximum number of consecutive elements $X[j]$ immediately preceding $X[i]$ such that $X[j] \leq X[i]$

Spans have applications to financial analysis, e.g., stock at 52-week high

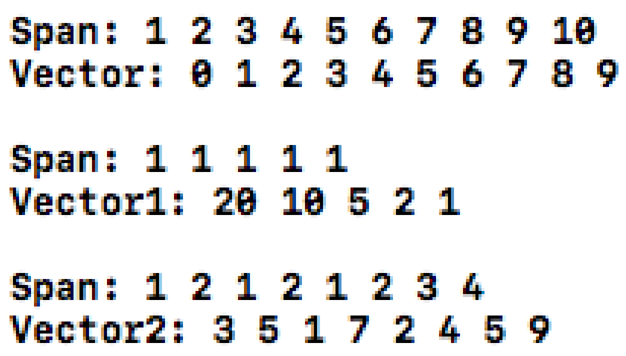
1. Example of computing the spans S of X.

X	6	3	4	5	2
S	1	1	2	3	1

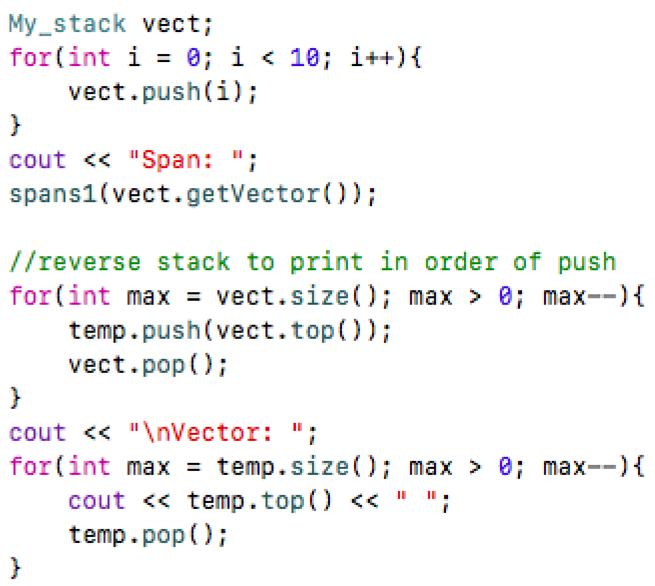
2. (10 pts) Compute the spans based on the definition above:

```
Algorithm spans1(x)
Input: vector x of n integers
Output: vector s of spans of the vector x
for i = 0 to n-1 do
   j = 1
   while (j <= i ∧ x[i-j] <= x[i])
      j =  j + 1
   s[i] = j
return s
```

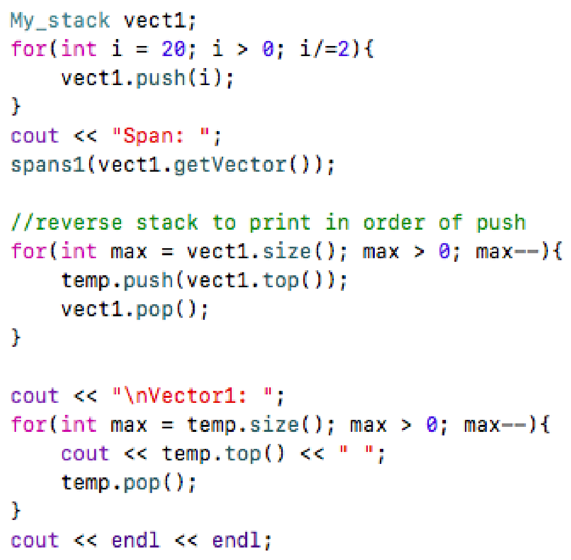
1. (10 pts) Test the algorithm above for correctness using at least three different input vectors.



```
Span: 1 2 3 4 5 6 7 8 9 10
Vector: 0 1 2 3 4 5 6 7 8 9

Span: 1 1 1 1 1
Vector1: 20 10 5 2 1

Span: 1 2 1 2 1 2 3 4
Vector2: 3 5 1 7 2 4 5 9
```



```
My_stack vect;
for(int i = 0; i < 10; i++){
    vect.push(i);
}
cout << "Span: ";
spans1(vect.getVector());

//reverse stack to print in order of push
for(int max = vect.size(); max > 0; max--){
    temp.push(vect.top());
    vect.pop();
}
cout << "\nVector: ";
for(int max = temp.size(); max > 0; max--){
    cout << temp.top() << " ";
    temp.pop();
}
```



```
My_stack vect1;
for(int i = 20; i > 0; i/=2){
    vect1.push(i);
}
cout << "Span: ";
spans1(vect1.getVector());

//reverse stack to print in order of push
for(int max = vect1.size(); max > 0; max--){
    temp.push(vect1.top());
    vect1.pop();
}

cout << "\nVector1: ";
for(int max = temp.size(); max > 0; max--){
    cout << temp.top() << " ";
    temp.pop();
}
cout << endl << endl;
```

```
My_stack vect2;
vect2.push(3);
vect2.push(5);
vect2.push(1);
vect2.push(7);
vect2.push(2);
vect2.push(4);
vect2.push(5);
vect2.push(9);

cout << "Span: ";
spans1(vect2.getVector());

//reverse stack to print in order of push
for(int max = vect2.size(); max > 0; max--){
    temp.push(vect2.top());
    vect2.pop();
}
cout << "\nVector2: ";
for(int max = temp.size(); max > 0; max--){
    cout << temp.top() << " ";
    temp.pop();
}
cout << endl << endl;
```

2. (10 pts) What is the running time function of the algorithm above? The function should define the relation between the input size and the number of comparisons perform on elements of the vector. How can you classify the algorithms using the Big-O asymptotic notation and why?

   - The running time function of the spans1 is $f(n) = 2n^2 + 1$. This algorithm is $O(n^2)$ because the of the nested while loop inside of a for loop which combined results in $n * n$ comparisons.

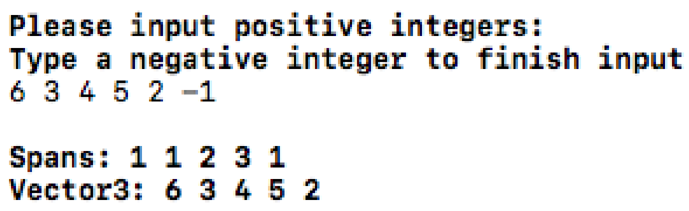
```
vector<int> spans1(vector<int> x){
    vector<int> s;
    s.resize(x.size());
    for(int i = 0; i < x.size(); i++){
        int j = 1;
        while(j <= i && x[i-j] <= x[i] && x[i-j+1] >= x[i-j]){
            j = j + 1;
        }
        s[i] = j;
        cout << s[i] << " ";
    }
    return s;
}
```

3. (20 pts) Compute the spans using a stack as an auxiliary data structure storing (some) indexes of x.

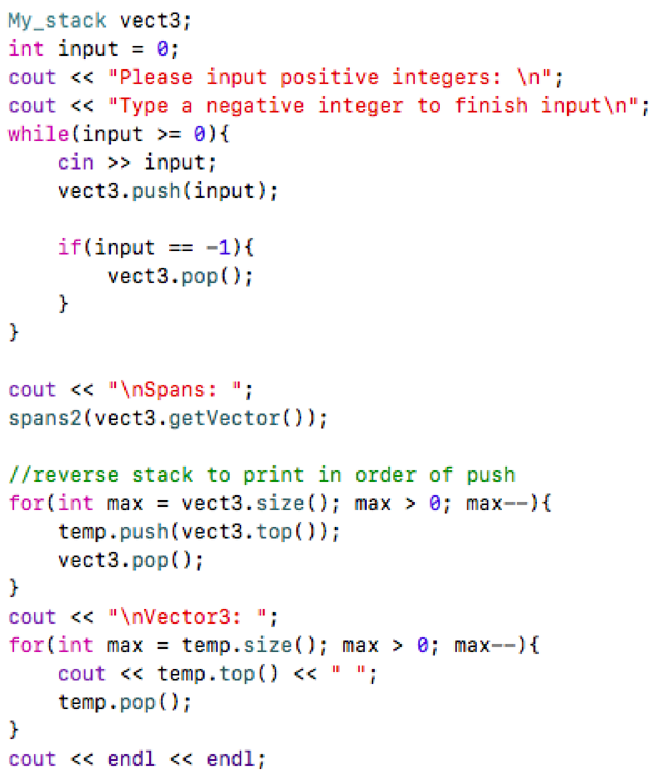
   Algorithm spans2:

   (a) We scan the vector x from left to right
   (b) Let i be the current index
   (c) Pop indices from the stack until we find index j such that x[i] $<$ x[j] is true
   (d) Set s[i] $=$ i $-$ j
   (e) Push i onto the stack

4. (10 pts) Test the second algorithm (spans2) for correctness using at least three different input vectors. Your program should run correctly on TA's input.



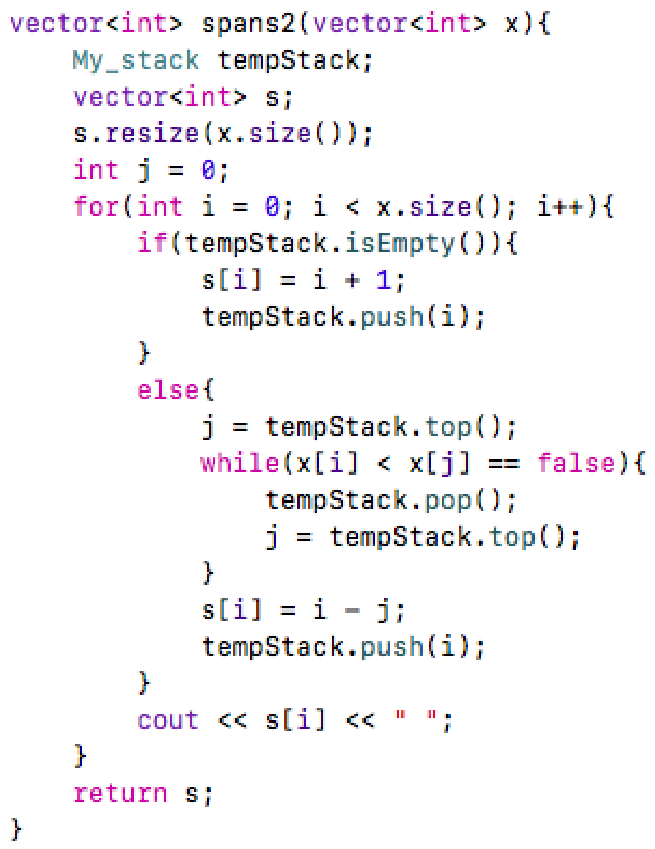
```
Please input positive integers:
Type a negative integer to finish input
6 3 4 5 2 -1

Spans: 1 1 2 3 1
Vector3: 6 3 4 5 2
```



```
My_stack vect3;
int input = 0;
cout << "Please input positive integers: \n";
cout << "Type a negative integer to finish input\n";
while(input >= 0){
    cin >> input;
    vect3.push(input);

    if(input == -1){
        vect3.pop();
    }
}

cout << "\nSpans: ";
spans2(vect3.getVector());

//reverse stack to print in order of push
for(int max = vect3.size(); max > 0; max--){
    temp.push(vect3.top());
    vect3.pop();
}
cout << "\nVector3: ";
for(int max = temp.size(); max > 0; max--){
    cout << temp.top() << " ";
    temp.pop();
}
cout << endl << endl;
```

5. (10 pts) What is the running time function of the second algorithm? The function should define the relation between the input size and the number of comparisons perform on elements of the vector. How can you classify the algorithms using the Big-O asymptotic notation and why? Compare the performance of both the algorithms.

- The running time function of the spans1 is $f(n) = 2n+1$. This algorithm is $O(n)$ because the of the nested loops which combined results in $n * n$ comparisons. Both algorithms have the same asymptotic notation, however the second algorithm uses less memory to compare the vector elements and ensures that the span count only concerns with consecutive numbers.



```
vector<int> spans2(vector<int> x){
    My_stack tempStack;
    vector<int> s;
    s.resize(x.size());
    int j = 0;
    for(int i = 0; i < x.size(); i++){
        if(tempStack.isEmpty()){
            s[i] = i + 1;
            tempStack.push(i);
        }
        else{
            j = tempStack.top();
            while(x[i] < x[j] == false){
                tempStack.pop();
                j = tempStack.top();
            }
            s[i] = i - j;
            tempStack.push(i);
        }
        cout << s[i] << " ";
    }
    return s;
}
```

6. (10 pts) Programming style, and program organization and design: naming, indentation, whitespace, comments, declaration, variables and constants, expressions and operators, line length, error handling and reporting, files organization. Please refer to the PPP-style document.

7. Instructions

   (a) Your files should be arranged as below

      i. Declaration of `My_stack` class in `My_stack.h`

      ii. Definition (implementation) of `My_stack` class in `My_stack.cpp`

      iii. Algorithm implementations and the `main()` function in the file `Application.cpp`

   (b) Compile your program by

      ```
      g++ -std=c++11 *.cpp
      ```

      or

      ```
      make all
      ```

   (c) Run your program by executing

      ```
      ./Main
      ```

   (d) Testing code in `Application.cpp` and collect output data for your report.

8. Submission

   (a) Tar the files above and any extra files. Please create your tar file following the instructions.

   (b) "turnin" your tar file to eCampus no later than **July** 18.

   (c) (20 points) Submit a hard copy of cover page, report (see below), `My_stack.h`, `My_stack.cpp` and `Application.cpp` in lab to your lab TA. Find a cover page .

      i. Typed report made preferably using "LYX/LaTeX"

         A. (2 pts) Program description; purpose of the assignment

         B. (4 pts) Data structures description

            - Theoretical definition
            - Real implementation
            - Analysis of the best and worst scenarios for computing spans.

      ii. (2 pts) Instructions to compile and run your program; input and output specifications

      iii. (2 pts) Logical exceptions (and bug descriptions)

      iv. (5 pts) C++ object oriented or generic programming features, C++11 features

      v. (5 pts) Testing results