Due July 20 at midnight to eCampus

First Name Suqian Last Name Wang UIN 825009505

User Name wangsuqian123 E-mail address wangsuqian123@tamu.edu

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

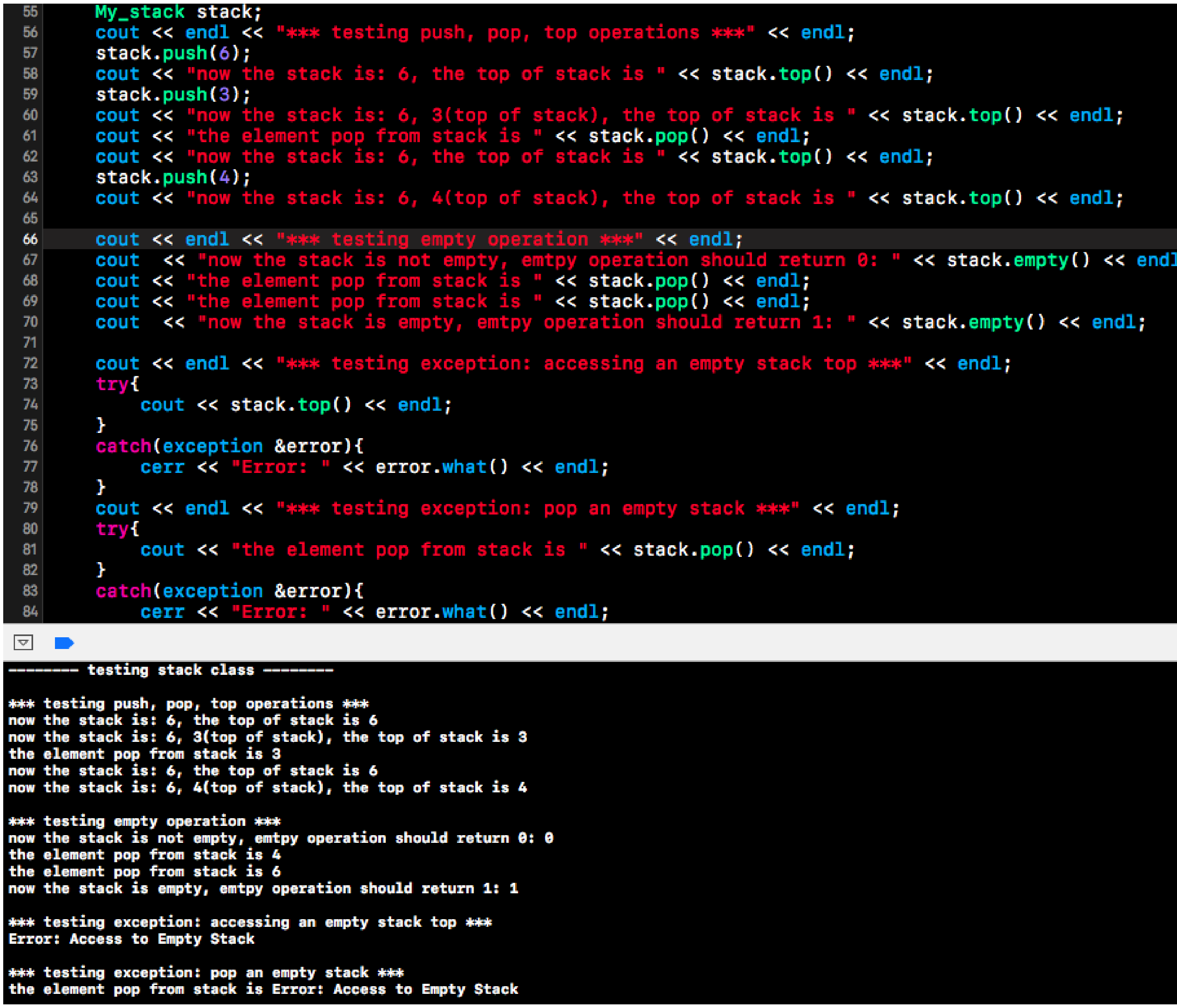
Type of sources	lecture slides	website	website
People	Teresa Leyk		
Web pages (provide URL)		http://en.cppreference.com/w/cpp/language/range-for	https://en.wikipedia.org/wiki/C%2B%2B11
Printed material			
Other Sources			

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 Suqian Wang Date July 20, 2017

Programming Assignment 2 (140 points)

1. (30 pts) Implement a stack ADT using STL vector to support the operations: `push()`, `pop()`, `top()`, `empty()`.
2. (10 pts) Test the operations of the stack class for correctness.



```cpp
    My_stack stack;
    cout << endl << "*** testing push, pop, top operations ***" << endl;
    stack.push(6);
    cout << "now the stack is: 6, the top of stack is " << stack.top() << endl;
    stack.push(3);
    cout << "now the stack is: 6, 3(top of stack), the top of stack is " << stack.top() << endl;
    cout << "the element pop from stack is " << stack.pop() << endl;
    cout << "now the stack is: 6, the top of stack is " << stack.top() << endl;
    stack.push(4);
    cout << "now the stack is: 6, 4(top of stack), the top of stack is " << stack.top() << endl;

    cout << endl << "*** testing empty operation ***" << endl;
    cout  << "now the stack is not empty, emtpy operation should return 0: " << stack.empty() << endl
    cout << "the element pop from stack is " << stack.pop() << endl;
    cout << "the element pop from stack is " << stack.pop() << endl;
    cout  << "now the stack is empty, emtpy operation should return 1: " << stack.empty() << endl;

    cout << endl << "*** testing exception: accessing an empty stack top ***" << endl;
    try{
        cout << stack.top() << endl;
    }
    catch(exception &error){
        cerr << "Error: " << error.what() << endl;
    }
    cout << endl << "*** testing exception: pop an empty stack ***" << endl;
    try{
        cout << "the element pop from stack is " << stack.pop() << endl;
    }
    catch(exception &error){
        cerr << "Error: " << error.what() << endl;
```

```
-------- testing stack class --------

*** testing push, pop, top operations ***
now the stack is: 6, the top of stack is 6
now the stack is: 6, 3(top of stack), the top of stack is 3
the element pop from stack is 3
now the stack is: 6, the top of stack is 6
now the stack is: 6, 4(top of stack), the top of stack is 4

*** testing empty operation ***
now the stack is not empty, emtpy operation should return 0: 0
the element pop from stack is 4
the element pop from stack is 6
now the stack is empty, emtpy operation should return 1: 1

*** testing exception: accessing an empty stack top ***
Error: Access to Empty Stack

*** testing exception: pop an empty stack ***
the element pop from stack is Error: Access to Empty Stack
```

1. What is the running time of each operation? Express it using the Big-O asymptotic notation.

   Running time for each operation:

   push(): One .push_back operation. f(n) = 1. Classification = $O(1)$

   pop(): One comparison to see if the stack is empty and one .pop_back operation. f(n) = 2. Classification = $O(1)$

   top(): One comparison to see if the stack is empty and one .back operation. f(n) = 2. Classification = $O(1)$

   empty(): One comparison to see if the stack is empty. f(n) = 1. Classification = $O(1)$

1. 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

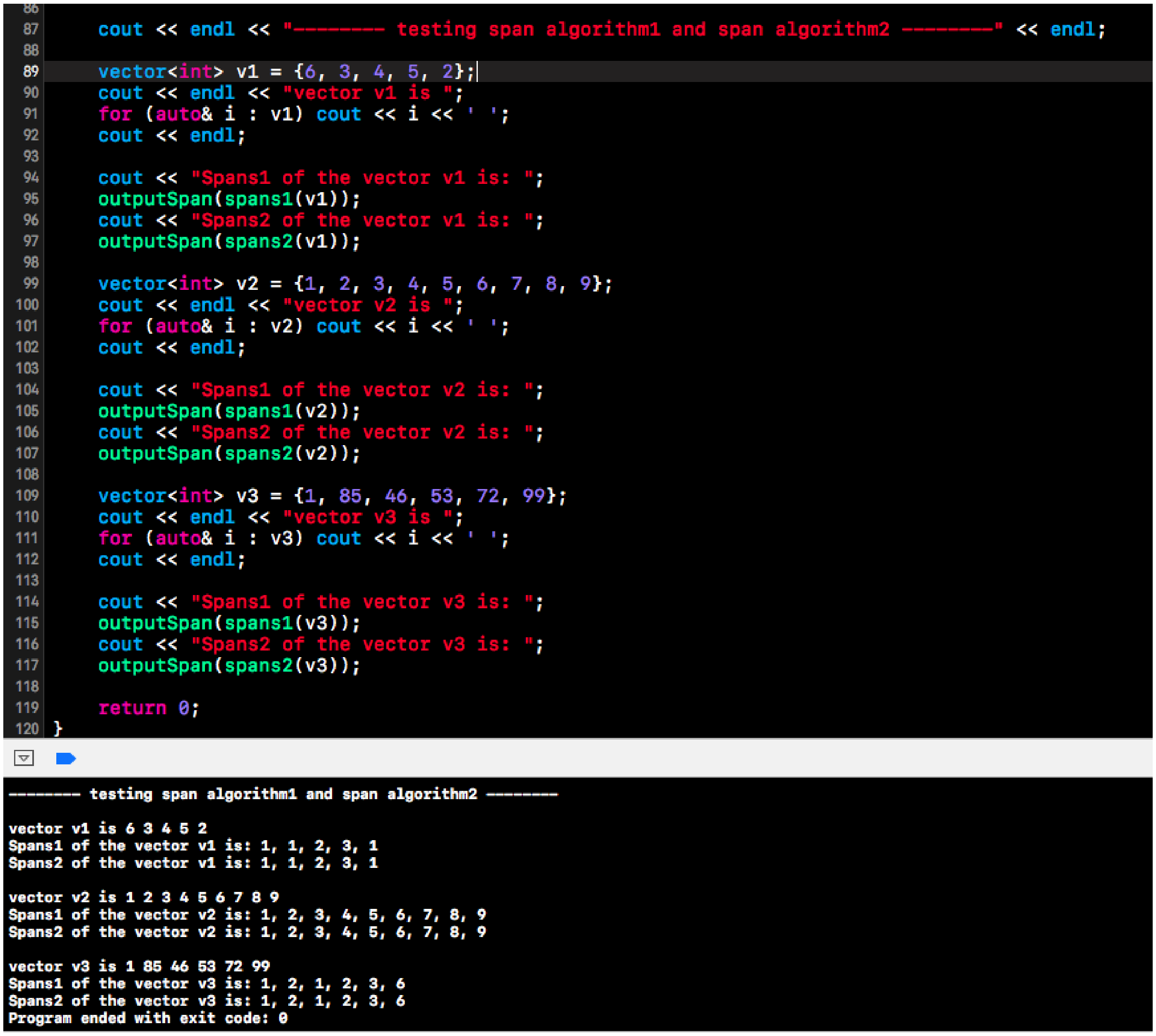
1. Example of computing the spans S of X.

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.



```
    cout << endl << "-------- testing span algorithm1 and span algorithm2 --------" << endl;

    vector<int> v1 = {6, 3, 4, 5, 2};
    cout << endl << "vector v1 is ";
    for (auto& i : v1) cout << i << ' ';
    cout << endl;

    cout << "Spans1 of the vector v1 is: ";
    outputSpan(spans1(v1));
    cout << "Spans2 of the vector v1 is: ";
    outputSpan(spans2(v1));

    vector<int> v2 = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    cout << endl << "vector v2 is ";
    for (auto& i : v2) cout << i << ' ';
    cout << endl;

    cout << "Spans1 of the vector v2 is: ";
    outputSpan(spans1(v2));
    cout << "Spans2 of the vector v2 is: ";
    outputSpan(spans2(v2));

    vector<int> v3 = {1, 85, 46, 53, 72, 99};
    cout << endl << "vector v3 is ";
    for (auto& i : v3) cout << i << ' ';
    cout << endl;

    cout << "Spans1 of the vector v3 is: ";
    outputSpan(spans1(v3));
    cout << "Spans2 of the vector v3 is: ";
    outputSpan(spans2(v3));

    return 0;
}
```

```
-------- testing span algorithm1 and span algorithm2 --------

vector v1 is 6 3 4 5 2
Spans1 of the vector v1 is: 1, 1, 2, 3, 1
Spans2 of the vector v1 is: 1, 1, 2, 3, 1

vector v2 is 1 2 3 4 5 6 7 8 9
Spans1 of the vector v2 is: 1, 2, 3, 4, 5, 6, 7, 8, 9
Spans2 of the vector v2 is: 1, 2, 3, 4, 5, 6, 7, 8, 9

vector v3 is 1 85 46 53 72 99
Spans1 of the vector v3 is: 1, 2, 1, 2, 3, 6
Spans2 of the vector v3 is: 1, 2, 1, 2, 3, 6
Program ended with exit code: 0
```

1. (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 worst case is when the vector is sorted in ascending order. The outer for loop executes n times, for each iteration, there will be two assignment operations. The while loop executes $n(n-1)/2$ times, for each iteration, there are two operation inside the while loop, one assignment and one addition. Running time function $f(n) = 2n + n(n-1) = n^2 + n$, classification $= O(n^2)$

1. (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

2. (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.

   Test case and result were include in the previous screenshot.

1. (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 worst case is when all elements are sorted in decreasing order. Every element of array is added and removed from stack once, so there will be total 2n operations. Running time funciton $f(n) = 2n$, classification $= O(n)$

(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.

Program description

This program implements a stack ADT using STL vector to support the operations: push(), pop(), top(), empty(). Then the program use the implemented stack class to compute spans. This program also compute the spans without class by operating on vector directly.

purpose of the assignment

The purpose of the assignment is to define a stack class using vector and use the defined stack to find the span in a given vector. Also, compare the complexity of the two spans algorithm.

Data structures description

Theoretical definition

Abstract Data Type (ADT) specifies

the type of the data stored
the operations that support the data

The main feature of ADT is

a clear description of the input to each operation
the action of each operation
its return type.

Real implementation

The type of the data stored

A user-defined class called My_stack. This class uses a vector to store elements and performs the stack operations on data as well.
Vector. Used in class as a private member, and used in main.cpp directly for Algorithm spans1 and testing.

The operations that support the data

There are four operations on the stack: push(), pop(), top(), and empty(). push() pushes element onto the stack, pop() decrements the number of elements on stack and returns the top element, top() returns the top element on stack, empty() returns true if the stack is empty.
There are several operations on the vector as well. push_back() adds an element to a vector, back() returns the last element of a vector, pop_back() removes the last element of a vector, size() gets the number of the element in a vector.

Analysis of the best and worst scenarios for computing spans.

For spans1 algorithm

the best case is the vector is sorted in descending order, the while loop will never be executed, there will be $2n$ operations total, classification $= O(n)$.
the worst case is the vector is sorted in ascending order, as I mentioned befor, there will be $n^2 + 2n$ operations total, classification $= O(n^2)$.

For spans2 algorithm

the best case is the vector is sorted in ascending order, every element will be pushed onto stack but there will be no pop operation, thus, there will be n operations total, classification $= O(n)$.
the worst case is the vector is sorted in descending order, every element will be pushed and poped from stack once, thus, there will be $2n$ operations total, classification $= O(n)$.

Instructions to compile and run your program

compile: g++ -std=c++11 *.cpp -o Main

run: ./Main

input and output specifications

Input: there is no terminal input because all test cases were written in Application.cpp.

Output: output is supposed to show the result of each test case

- push element on stack one by one and display the elements on stack every time by calling top()
- pop element on stack and display the poped element, display the top element() afterwards to show the stack status after pop
- display the boolean value of empty()
- exception should be caught and displayed because of accessing the top of an empty stack or pop an empty stack
- given a vector, output the results after implement each spans algorithm on this vector

Logical exceptions (and bug descriptions)

There are exceptions throughtout the program

- Accessing the top of an empty stack should throw an error.
- Poping element from an empty stack should throw an error.

C++ object oriented or generic programming features, C++11 features

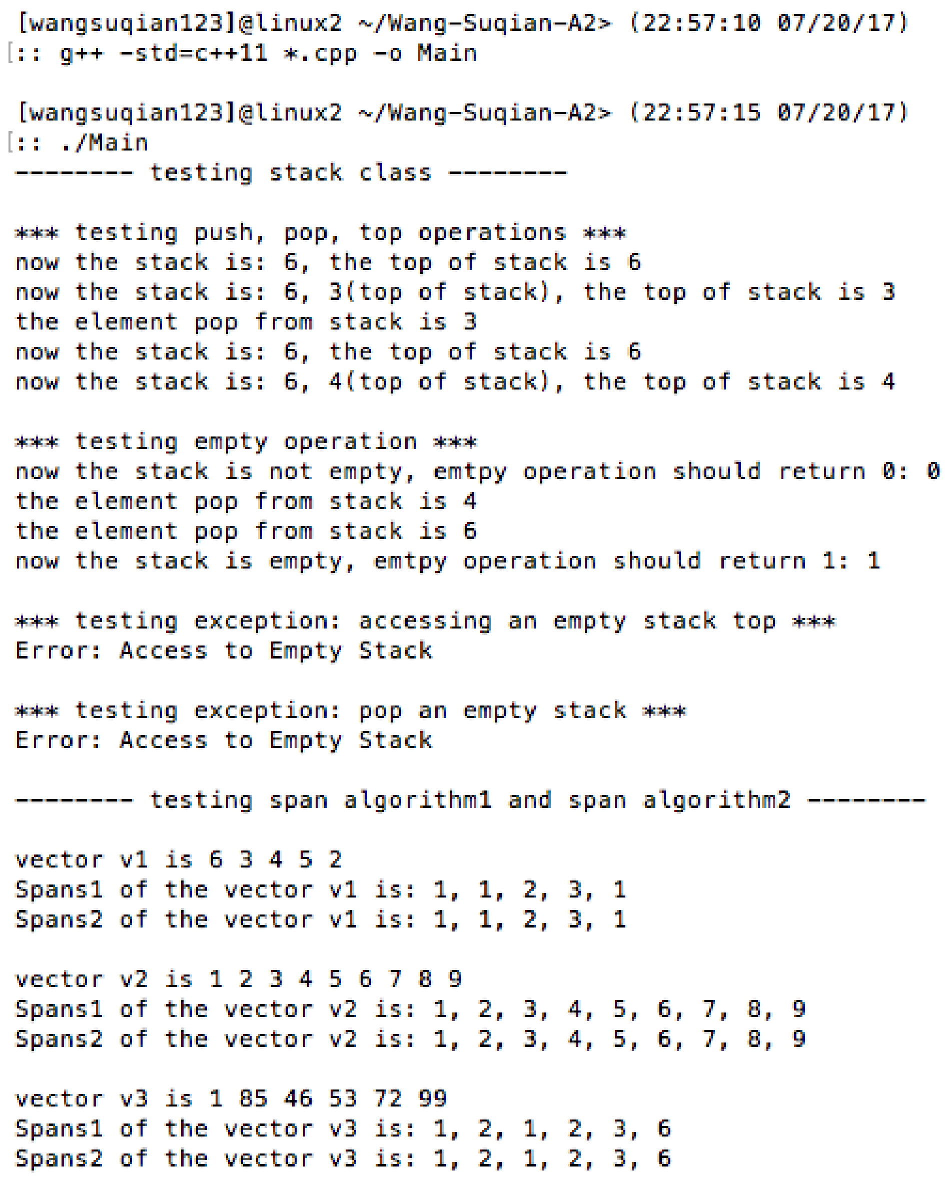
My_stack is a C++ object oriented feature, including the variables of My_stack type defined in Application.cpp

No generic programming feature because I didn't use a template since all the elements are int type throughout the program

C++11 features

- range-based for loop: for example, for (auto i : v)
- uniform initialization syntax: c++11 vector container initializer can avoid push_back by using braces and equal sign. For example, vector<int> = {1, 2, 3}

Testing results



```
[wangsuqian123]@linux2 ~/Wang-Suqian-A2> (22:57:10 07/20/17)
[:: g++ -std=c++11 *.cpp -o Main

[wangsuqian123]@linux2 ~/Wang-Suqian-A2> (22:57:15 07/20/17)
[:: ./Main
-------- testing stack class --------

*** testing push, pop, top operations ***
now the stack is: 6, the top of stack is 6
now the stack is: 6, 3(top of stack), the top of stack is 3
the element pop from stack is 3
now the stack is: 6, the top of stack is 6
now the stack is: 6, 4(top of stack), the top of stack is 4

*** testing empty operation ***
now the stack is not empty, emtpy operation should return 0: 0
the element pop from stack is 4
the element pop from stack is 6
now the stack is empty, emtpy operation should return 1: 1

*** testing exception: accessing an empty stack top ***
Error: Access to Empty Stack

*** testing exception: pop an empty stack ***
Error: Access to Empty Stack

-------- testing span algorithm1 and span algorithm2 --------

vector v1 is 6 3 4 5 2
Spans1 of the vector v1 is: 1, 1, 2, 3, 1
Spans2 of the vector v1 is: 1, 1, 2, 3, 1

vector v2 is 1 2 3 4 5 6 7 8 9
Spans1 of the vector v2 is: 1, 2, 3, 4, 5, 6, 7, 8, 9
Spans2 of the vector v2 is: 1, 2, 3, 4, 5, 6, 7, 8, 9

vector v3 is 1 85 46 53 72 99
Spans1 of the vector v3 is: 1, 2, 1, 2, 3, 6
Spans2 of the vector v3 is: 1, 2, 1, 2, 3, 6
```