Experiment #2

Static Stacks

Student's Name:	
Semester:	Date:

Assessment:

Assessment Point	**Weight**	**Grade**
Methodology and correctness of results		
Discussion of results		
Participation		
Assessment Points' Grade:		

Comments:

Experiment #2:

Static Stacks in C++ Programming Language

Objectives:

1. To introduce the students with the concept of stacks
2. To implement static stacks
3. To implement different functions of stacks
4. To understand the disadvantages of array implementation of stacks

Discussion:

Stacks are abstract data types with LIFO (Last In First Out) type of working, where a new element is added at one end (top) and an element is removed from that end only.

The functions associated with stack are:
1. isempty() – Returns whether the stack is empty – Time Complexity : O(1)
2. isfull – Returns whether the stack is full – Time Complexity : O(1)
3. Push(g) – Adds the element 'g' at the top of the stack – Time Complexity: O(1)
4. Pop() – Deletes the top most element of the stack – Time Complexity : O(1)

Static Stack implementation

```
// this program for implementing a static stack using templates
// programmed by Dr.Aryaf Al-adwan
#include <iostream>
Using namespace std;
const int SIZE = 10;
template <class T>
class stack
{
  private:
          int tos;
          T array[SIZE];
public:
  stack()
  {
     tos = -1;
```

```
  }

  bool isEmpty()
  {
          if(tos==-1)
                  return true;
          else
                  return false;
  }

  bool isFull()
  {
          if(tos==SIZE)
                  return true;
          else
                  return false;
  }

  void push(T);
  T pop();
  T max();
};

template <class T>
void stack <T> ::push(T element)
   {
     if(isFull())
          {
       cout << "Stack is full.\n";
     }
          else
          {
           array[tos] = element;
```

```
            ++tos;
           }
   }

template <class T>
   T stack<T>::pop()
   {
     if(isEmpty()) {
       cout << "Stack is empty.\n";
       return 0; // return null on empty stack
     }
           else
           {
           tos--;
           return array[tos];
           }
   }

void main()
{
 stack <char> s1,s2;
  s1.push('a');
  s1.push('b');
  s1.push('c');
  s2.push('x');
  s2.push('y');
  s2.push('z');
  cout<<s1.pop()<<endl;
  cout<<s1.pop()<<endl;
  cout<<s1.pop()<<endl;
  cout<<s2.pop()<<endl;
  cout<<s2.pop()<<endl;
  cout<<s2.pop()<<endl;
```

```
  cout<<"content of the s1 after pushing and poping is:"<<endl;
  s1.print_stack();
  cout<<"\ncontent of the s1 after pushing and poping is:"<<endl;
  s2.print_stack();
  stack<double> ds1, ds2;
  ds1.push(1.1);
  ds1.push(3.3);
  ds1.push(5.5);
  ds2.push(2.2);
  ds2.push(4.4);
  ds2.push(6.6);
  cout<<ds2.max();
  cout<<endl;
  for(int i=0; i<3; i++)
     cout << "Pop ds1: " << ds1.pop() << "\n";
  for( i=0; i<3; i++)
     cout << "Pop ds2: " << ds2.pop() << "\n";
}
```

Exercise 1:

Write a c++ program to print the contents of the stack?

Solution to Exercise 1

Output

Exercise 2:

Write a c++ program to find the maximum element stored in the stack?

Solution to Exercise 2

Output

Exercise 3:

Write a c++ program to print the word " Welcome in FET" in revere order?

Solution to Exercise 3

Output

Exercise 4:

Write a program in C++ to separate integers stored in a stack into two stacks one for the even integers and the other for odd ones.

Solution to Exercise 3

Output