|  |  |
| --- | --- |
| **Data Structures & Algorithms**  Diploma in CSF, IT  Year 2/3 (2020/21) Semester 4/6 | **Week 4** |
| **1 Hour** |
| **Tutorial 4 – Stacks** | |

1. Suppose that s and t are empty stacks and a, b, c, and d are objects. What do the stacks contain after the following sequence of operations executes?

s.push(a);

s.push(b);

s.push(c);

t.push(d);

t.push(s.getTop());

s.pop();

t.push(s.getTop());

s.push(t.getTop());

t.pop();

t.pop();

|  |
| --- |
|  |

1. The specification of the Stack ADT implemented using Pointers is given below.

|  |
| --- |
| //Stack.h (Pointer-based implementation)  #pragma once  #include <iostream>  using namespace std;  typedef int ItemType;  class Stack  {  private:  struct Node  {  ItemType item;  Node \*next;  };  Node \*topNode;  public:  //Default constructor  Stack();  //Destructor  ~Stack();  //check if the stack is empty  bool isEmpty();  //push item on top of the stack  bool push(ItemType &item);  //pop item from top of stack  bool pop();  //retrieve and pop item from top of stack  bool pop(ItemType &item);  //retrieve item from top of stack  void getTop(ItemType &item);  //display items in stack in order  void displayInOrder();  //display items in stack in order of insertion  void displayInOrderOfInsertion();  }; |

Implement the following operations of the List ADT

1. **~Stack();**

|  |
| --- |
|  |

1. **bool pop(ItemType& item);**

|  |
| --- |
|  |

1. **void displayInOrder();** // without worry about changing the stack

|  |
| --- |
|  |

*Note : The stack is empty after the above is executed.*

1. **void displayInOrderOfInsertion();**

|  |
| --- |
|  |

1. Using a stack, implement the **reverse()** function that takes a string as an argument and returns the string with its characters reversed.

For example: **reverse("abcde")** returns **"edcba"**.

The function prototype is:

**string reverse(string);**