|  |  |
| --- | --- |
| Data Structures & Algorithms Diploma in IT, ISF  Year 2 (2017/18) Semester 4 | Week 4 |
| 2-3 Hours |
| **Practical 4 – Stacks** | |

**Objectives**

At the end of this practical, the students should be able to:

* Implement a Stack ADT
* Apply use of stacks in a simple application

|  |
| --- |
| **IMPORTANT**   * Upload all your work to DSA network folder **\\ictspace.ict.np.edu.sg\dsa**. by the designated timeline given by your tutor |

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

|  |
| --- |
| //stack.h  //array-based implementation  #ifndef STACK\_H  #define STACK\_H  #include <iostream>  #include <string>  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();  };  #endif |

NB: for displayInOrderOfInsertion(), the original stack should still contain the original contents after execution of the function even though some manipulation of the stack may be required to display the items

2. Implement the operations of the Stack ADT

*Note : You should implement (and test) one operation at a time.*

3. Write a sample program, StackDemo.cpp, to do the following:

1. Create an empty stack, s
2. Push ‘3’ on top of the stack
3. Push ‘4’ on top of the stack
4. Get top item of the stack and display
5. Display all the items in the stack in order of insertion
6. Pop top item from stack
7. Display all the items in the stack in order of insertion

4. Now improve StackDemo.cpp to do the following:

1. Read user input of an infix expression
2. Convert the infix expression to postfix form.
3. Evaluate the postfix expression and output the answer of the evaluation.

NB: you may need to make changes to Stack.h and/or Stack.cpp to accomplish the above question.

Additional hints: you **may** use some functions from the standard library such as isdigit() or isalnum()