Assignment :2

Name : Muhamad Shaheer

Roll no:20p-0480

Q1: Stack using Array

Code:

#include<iostream>

using namespace std;

class stack

{

 int \*arr;

 int size;

 int len;

 int\* curr;

 int top;

 public:

  stack()

  {

   size=10;

   len=0;

   arr=new int [size];

   top=-1;

  }

  void print()

  {

   while(!IsEmpty())

   {

    cout<<arr[top]<<" ";

    pop();

   }

   cout<<endl;

  }

  void push(int val)

  {

   if(top==size-1)

   {

    cout<<"Stack Overflow"<<endl;

   }

   else

   {

    top++;

    arr[top]=val;

    len++;

   }

  }

  int pop()

  {

   if (top==-1)

   {

    cout<<"Stack Underflow"<<endl;

   }

   else

   {

    int x=arr[top];

    top--;

    len--;

    return x;

   }

  }

  int peek()

  {

   if(top==-1)

   {

    cout<<"Stack Underflow"<<endl;

   }

   else

   {

    return arr[top];

   }

  }

  bool IsEmpty()

  {

   if (top==-1)

   {

    return true;

   }

   else

   {

    return false;

   }

  }

};

int main()

{

 stack s;

 cout<<"push elements are:"<<endl;

 s.push(1);

 s.push(2);

 cout<<"deleted element 2 from stack"<<endl;

 s.pop();

 s.push(3);

 s.push(4);

 s.push(5);

 cout<<endl;

 cout<<"peek element :"<<s.peek();

 cout<<endl;

 s.print();

 cout<<endl;

 return 0;

}

Output:



Q2: Stack using Linked List

Code: #include<iostream>

using namespace std;

class node

{

    public:

    node\*prev;

    node\*next;

    int data;

    node(int val)

    {

        data = val;

        prev=next=NULL;

    }

};

class Stack

{

    private:

    int noOfElements;

    int size;

    public:

    node\*top;

    Stack()

    {

        top = NULL;

        noOfElements = 0;

        size=10;

    }

    ~Stack() {

        while (top) {

            node\* temp = top;

            top = top->next;

            delete temp;

        }

    }

//declaring funtions

    void push(int val);

    void display();

    int pop();

    void reverseStack();

    int peek()

    {

        return top->data;

    }

    bool isEmpty();

};

int main()

{

    Stack s;

    cout<<"push elements are:"<<endl;

    s.push(1);

    s.push(2);

    s.push(3);

    s.push(4);

    s.push(5);

    s.push(6);

    s.display();

    cout<<"after removal "<<endl;

    s.pop();

    s.display();

    cout<<"reverese list is"<<endl;

    s.reverseStack();

    s.display();

    return 0;

}

bool Stack ::isEmpty()

{

    if(noOfElements == 0)

    return true;

    else

    return false;

}

int Stack ::pop()

{

    node\*curr = top;

    if(curr == NULL)

    {

        cout<<"List is empty: Stack Underflow ,return -1"<<endl;

        return -1;

    }

    else

    {

        node\*curr = top;

        int data = curr->data;

        top = curr->prev;

        delete curr;

        noOfElements--;

        return data;

    }

}

void Stack ::push(int val)

{

    if(noOfElements<size)

    {

        node\*t=new node(val);

        if(top == NULL)

        {

            top=t;

            noOfElements++;

        }

        else{

            t->prev=top;

            top=t;

            noOfElements++;

        }

    }

    else

    cout<<"Stack full : Overflow"<<endl;

}

void Stack::display()

{

    node\*curr=top;

    while(curr!=NULL)

    {

        cout<<curr->data<<endl;

        curr =curr->prev;

    }

}

void Stack ::reverseStack() {

    // use 3 variables to traverse list

        node\* prev = NULL;

        node\* curr = top;

        node\* next = NULL;

        while (curr) {

            // Store next in list

            next = curr->next;

            // Reverse list

            curr->next = prev;

            // Move the pointers

            prev = curr;

            curr = next;

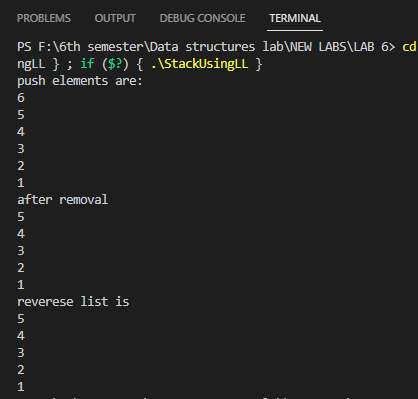
        }

        // Update top

        top = prev;

}

Output:



But it works fine on this implementation of stack

#include <iostream>

using namespace std;

struct Node {

    int data;

    Node\* next;

};

// Define the Stack class

class Stack {

public:

    Stack() : top(nullptr) {}

    ~Stack() {

        while (top) {

            Node\* temp = top;

            top = top->next;

            delete temp;

        }

    }

    void push(int data) {

        Node\* temp = new Node;

        temp->data = data;

        temp->next = top;

        top = temp;

    }

    // Pop the top element from the stack

    void pop() {

        if (!top) {

            cout << "Stack is empty!" << endl;

            return;

        }

        Node\* temp = top;

        top = top->next;

        delete temp;

    }

    bool isEmpty() {

        return (top == nullptr);

    }

    int peek() {

        if (top) {

            return top->data;

        } else {

            return -1;

        }

    }

    // Reverse the order of elements in the stack

    void reverse() {

        // use 3 variables to traverse list

        Node\* prev = NULL;

        Node\* curr = top;

        Node\* next = NULL;

        while (curr) {

            // Store next in list

            next = curr->next;

            // Reverse list

            curr->next = prev;

            // Move the pointers

            prev = curr;

            curr = next;

        }

        // Update top

        top = prev;

    }

    void printStack() {

        Node\* temp = top;

        while (temp) {

            cout << temp->data << " ";

            temp = temp->next;

        }

        cout << endl;

    }

private:

    Node\* top;

};

int main() {

    Stack s;

    s.push(1);

    s.push(2);

    s.push(3);

    s.push(4);

    s.push(5);

    s.printStack();

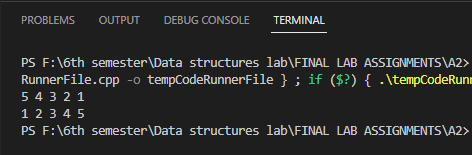
    s.reverse();

    s.printStack();

    return 0;

}

Output:



The End\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_