Practical 03

Abhishek Gupta 2019450017

1) Write a menu driven program to demonstrate Stack using array.

Code :

```
#include<iostream>
using namespace std;

class InfixToPostfix
{
    public:

    string stack[100],expression[100];
    string equation;
    int size,len,x,top;

    InfixToPostfix()
    {
        top=-1;
    }

    void get()
    {
        cout <<"Enter the Equation : "<<" ";
        getline(cin,equation);
        x=equation.length();
        equation.append(")",x);


        top++;
        stack[top]="(";
    }

    void operation()
    {
```

Stack Operations

```
        for(int i=0;i<x;i++)
        {

        }
    }

    void display()
    {
        cout<<"The Given Equation is : ";
        cout<<equation<<endl;
        cout<<"The Stack Array : ";
        for(int i=0;i<x+1;i++)
        {
            cout<<stack[i]<<" ";
        }
        cout<<endl;
        cout<<"The Expression Array : ";
        for(int i=0;i<x+1;i++)
        {
            cout<<expression[i]<<" ";
        }
        cout<<endl;
    }
};

int main()
{
    InfixToPostfix i;
    i.get();
    i.display();
    return 0;
}
```

Stack Operations

Practical 03

Abhishek Gupta 2019450017

Output :

```
C:\Users\gupta\Desktop\Stack Practical 03>s.exe
Enter size of stack : 10

Enter your choice :
1. Push
2. Pop
3. Display
4. Exit

Enter choice : 1

Enter element : 22
22 entered at index 0
Enter your choice :
1. Push
2. Pop
3. Display
4. Exit

Enter choice : 1

Enter element : 54
54 entered at index 1
Enter your choice :
1. Push
2. Pop
3. Display
4. Exit

Enter choice : 3

1  :  54
0  :  22
```

Stack Operations

Practical 03

Abhishek Gupta 2019450017

```
Enter your choice :
1. Push
2. Pop
3. Display
4. Exit

Enter choice : 2

54 removed from stack
Enter your choice :
1. Push
2. Pop
3. Display
4. Exit

Enter choice : 3

0  :  22

Enter your choice :
1. Push
2. Pop
3. Display
4. Exit

Enter choice : 4


C:\Users\gupta\Desktop\Stack Practical 03>
```

Stack Operations

Practical 03

Abhishek Gupta 2019450017

2) Write a program to evaluate Postfix expression using Stack.

Code :

```
#include<iostream>
#include<string.h>
#include<math.h>
using namespace std;

class PostFixEval
{
    private:
    int stack[100];
    string eq;
    int top,len,value,a,b,ans;
    public:
    PostFixEval()
    {
        top=-1;
    }

    void get()
    {
        cout<<"Enter String (Equation) : "<<" ";
        getline(cin,eq);
        len=eq.length();

        for(int i=0;i<len;i++)
        {
            if(eq[i] == '+')
            {
                a=pop();
                b=pop();
                ans=b+a;
```

Stack Operations

```
            cout<<"Addition is : "<<ans<<endl;
            push(ans);
        }
        else if(eq[i] == '-')
        {
            a=pop();
            b=pop();
            ans=b-a;
            cout<<"Subraction is : "<<ans<<endl;
            push(ans);
        }
        else if(eq[i] == '/')
        {
            a=pop();
            b=pop();
            ans=b/a;
            cout<<"Division is : "<<ans<<endl;
            push(ans);
        }
        else if(eq[i] == '*')
        {
            a=pop();
            b=pop();
            ans=b*a;
            cout<<"Multiplication is : "<<ans<<endl;
            push(ans);
        }
        else
        {
            value=eq[i]-48;
            push(value);
        }
```

Stack Operations

```
        }
    }

    void push(int value)
    {
        top++;
        stack[top]=value;
        cout<<value<<" Pushed ";
    }

    int pop()
    {
        int val=stack[top];
        top--;
        return val;
    }

    void display()
    {
        cout<<endl;
        cout<<"The Answer of given Equation is : "<<stack[top];
    }
};

int main()
{
    PostFixEval p;
    p.get();
    p.display();
    return 0;
}
```

Stack Operations

Output :

```
PS C:\Users\gupta\Desktop> ./f.exe
Enter String (Equation) :  231*+9-
2 Pushed 3 Pushed 1 Pushed Multiplication is : 3
3 Pushed Addition is : 5
5 Pushed 9 Pushed Subraction is : -4
-4 Pushed
The Answer of given Equation is : -4
PS C:\Users\gupta\Desktop>
```

3) Write a program to demonstrate Stack application (Infix to postfix expression conversion using Stack).

Code :

```
#include <iostream>
#include <string>
using namespace std;
class stack {
    int top1, top2, size;
    char s1[100], s2[100], var;
    string s;

public:
    stack()
    {
        top1 = top2 = -1;
    }
    void geteq()
    {
        cout << "\nEnter equation:- ";
```

Stack Operations

```
        getline(cin, s);
    }
    void push(char p)
    {
        top1++;
        s1[top1] = p;
    }
    void push2(char p)
    {
        top2++;
        s2[top2] = p;
    }
    char pop()
    {
        var = s1[top1];
        top1--;
        return var;
    }
    int prec(char p)
    {
        if (p == '+')
            return 0;
        if (p == '-')
            return 0;
        if (p == '*')
            return 1;
        if (p == '/')
            return 1;
        if (p == '^')
            return 2;
    }
    void optr(char p)
```

Stack Operations

```
    {
        char var1, var2;
        var1 = pop();
        if (var1 == '+' || var1 == '-' || var1 == '*' || var1 ==
'/' || var1 == '^') {
            if (prec(p) > prec(var1)) {
                push(var1);
                push(p);
            }
            else {
                push2(var1);
                push(p);
            }
        }
        else {
            push(var1);
            push(p);
        }
    }
    void close()
    {
        char var;
        var = pop();
        while (var != '(') {
            push2(var);
            var = pop();
        }
    }
    void comp()
    {
        int i = 0;
        push('(');
```

Stack Operations

```
        while (s[i] != '\0') {
            if (s[i] == '(')
                push(s[i]);
            else if (s[i] >= 'A' && s[i] <= 'Z')
                push2(s[i]);
            else if (s[i] >= 'a' && s[i] <= 'z')
                push2(s[i]);
            else if (s[i] == '+' || s[i] == '*' || s[i] == '-'
|| s[i] == '/' || s[i] == '^')
                optr(s[i]);
            else if (s[i] == ')')
                close();
            i++;
        }
        cout << "\nExpression:-";
        i = 0;
        while (i <= top2) {
            cout << s2[i];
            i++;
        }
    }
};
int main()
{
    stack s;
    s.geteq();
    s.comp();
}
```

Stack Operations

Output :

```
C:\Users\gupta\Desktop\Linked List 2>g++ infitopost.cpp -o i.exe

C:\Users\gupta\Desktop\Linked List 2>i.exe

Enter equation:- a+(b*c-(d/e^f)*g)*h)

Expression:-abc*def^/g*-h*+
C:\Users\gupta\Desktop\Linked List 2>
```

4) Write a program to demonstrate Stack application (Balancing parenthesis using Stack).

Code :

```
#include <iostream>
#include <string>
using namespace std;

class StackOperations
{
    char stack_arr[25];
    string equation;
    int top, x;

public:
    StackOperations()
    {
        top = -1;
    }

    void operation()
    {
        cout << "Enter the Equation : "
```

Stack Operations

```
            << " ";
        getline(cin, equation);
        x = equation.length();

        cout << "The Given Equation is :" << endl;
        for (int i = 0; i < x; i++)
        {
            cout << equation[i] << "  ";
        }

        for (int i = 0; i < x; i++)
        {
            if (equation[i] == '(')
            {
                push(equation[i]);
            }

            if (equation[i] == ')')
            {
                if (top == -1)
                {
                    cout << "Equation  is not Balanced";
                    goto xyz;
                }
                else
                {
                    pop();
                }
            }
        }

        compare();
```

Stack Operations

```
    xyz:
        cout << endl;
    }

    void push(char c)
    {
        top = top + 1;
        stack_arr[top] = c;
    }

    void pop()
    {
        if (top > -1)
        {
            stack_arr[top];
            top = top - 1;
        }
    }

    void compare()
    {
        if (top == -1)
        {
            cout << "Equation is balanced";
        }

        else
        {
            cout << "Equation is not balanced";
        }
    }
};
```

Stack Operations

Practical 03

Abhishek Gupta 2019450017

```
int main()
{
    StackOperations s;
    s.operation();
    return 0;
}
```

Output :

Command Prompt

```
C:\Users\gupta\Desktop>g++ Balance.cpp -o b.exe

C:\Users\gupta\Desktop>b.exe
Enter the Equation :  A*(c+d)
The Given Equation is :
A  *  (  c  +  d  )  Equation is balanced

C:\Users\gupta\Desktop>b.exe
Enter the Equation :  A+(c
The Given Equation is :
A  +  (  c  Equation is not balanced

C:\Users\gupta\Desktop>
```

Stack Operations

Practical 03

Abhishek Gupta 2019450017

5) Write a menu driven program to demonstrate Stack using linked list
Code :

```
#include<iostream>
#include<stdlib.h>
using namespace std;

struct node
{
    int data;
    struct node *next;
}
*list=NULL,*p,*s,*q,*r,*top=NULL;

class StackUsingLinkedList
{
public:
int choice,value;

void get()
{
    do
            {
                cout<<"0.Exit\n1.Push Operation\n2.Pop
Operation\n3.Display\n";
                cout<<"Enter Your Choice : "<<" ";
                cin>>choice;
                switch(choice)
                {
                    case 0:
                    break;
```

Stack Operations

```
                    case 1:
                        push_op();
                        break;

                    case 2:
                        pop_op();
                        break;

                    case 3:
                        display();
                        break;

                    default:
                        cout<<"invalid input"<<endl<<endl;
                }
            }while(choice!=0);

}

void push_op()
{
            cout<<"Enter the value : ";
            cin>>value;
            p=(struct node*)malloc(sizeof(node));
            p->data=value;
            if(list == NULL)
            {
                    list=p;
                    p->next=top;
                    top=p;
                    display();
            }
```

Stack Operations

```
            else if (!p)
            {
                cout<<"Overflow (Memory Full) ";
            }
            else
            {
                top->next=p;
                p->next=NULL;
                top=p;
                display();
            }
}

void pop_op()
{
            if(list == NULL)
            {
                cout<<"Under Flow (No Elements) ";
            }
            else
            {
                q=list;
                    while(q->next != NULL)
                {
                    r=q;
                    q=q->next;
                }
                top=r;
                top->next=NULL;
                delete q;
                display();
            }
```

Stack Operations

```
}

void display()
{
    if(list==NULL)
            {
                cout<<endl<<"List is Empty "<<endl<<endl;
            }
            else
            {
                cout<<"The List is : ";
                q=list;
                while(q !=NULL)
                {
                    cout<<q->data<<"|----->";
                    q=q->next;
                }
                cout<<endl<<endl;
            }
}
};

int main()
{
    StackUsingLinkedList s;
    s.get();
    return 0;
}
```

Output :

Stack Operations

Practical 03

Abhishek Gupta 2019450017

//Push

```
C:\Users\gupta\Desktop\Linked List 2>sl.exe
0.Exit
1.Push Operation
2.Pop Operation
3.Display
Enter Your Choice :  1
Enter the value : 10
The List is : 10|----->

0.Exit
1.Push Operation
2.Pop Operation
3.Display
Enter Your Choice :  1
Enter the value : 20
The List is : 10|----->20|----->

0.Exit
1.Push Operation
2.Pop Operation
3.Display
Enter Your Choice :  1
Enter the value : 30
The List is : 10|----->20|----->30|----->

0.Exit
1.Push Operation
2.Pop Operation
3.Display
Enter Your Choice :  1
Enter the value : 40
The List is : 10|----->20|----->30|----->40|----->
```

Stack Operations

//Pop

```
0.Exit
1.Push Operation
2.Pop Operation
3.Display
Enter Your Choice :  2
The List is : 10|----->20|----->30|----->

0.Exit
1.Push Operation
2.Pop Operation
3.Display
Enter Your Choice :  2
The List is : 10|----->20|----->

0.Exit
1.Push Operation
2.Pop Operation
3.Display
Enter Your Choice :  0

C:\Users\gupta\Desktop\Linked List 2>
```

Stack Operations