Module 3 : Stacks

1. Stacks using array

```
#include<iostream.h>
#include<conio.h>
#define MAX 10
int st[MAX];
int top=-1;
void push(int st[],int val);
void pop(int st[]);
void peep(int st[]);
void display(int st[]);

int stack[100],choice,n,x,i;
void push(void);
void pop(void);
void peep(void);
void main(){
        clrscr();
        top=-1;
        cout<<"\nEnter size of stack:";
        cin>>n;
        cout<<"\nStack operation :\n1.Push \n2.Pop \n3.Peep \n4.Exit";
        do{
                cout<<"\nEnter your choice";
                cin>>choice;
                switch(choice){
                        case 1:push();
                        break;
                        case 2:pop();
                        break;
                        case 3:peep();
                        break;
                        case 4:cout<<"exit";
                        break;
                        default:cout<<"Wrong choice";
                        break;
                }
        }while(choice!=4);
        getch();
}
void push(){
        if(top==MAX-1){
                cout<<"\nStack is overflow";
        }
        else{
                cout<<"Enter the element to insert";
                cin>>x;
```

```
                top++;
                stack[top]=x;
        }
}
void pop(){
        if(top<=-1)
        {
                cout<<"\nStack is underflow";
        }
        else
        {
                cout<<"\nPopped element is"<<stack[top];
                top--;
        }
}
void peep()
{
        if(top>=0){
                cout<<"Elements in the stack are:";

                for(i=0;i<=top;i++){
                        cout<<stack[i]<<"  ";
                }

        }
        else{
                cout<<"\nStack is empty";
        }
}

 /* Output:-
Enter size of stack:8

Stack operation :
1.Push
2.Pop
3.Peep
4.Exit
Enter your choice1
Enter the element to insert2

Enter your choice1
Enter the element to insert4

Enter your choice1
Enter the element to insert5

Enter your choice3
Elements in the stack are:2  4  5
Enter your choice2
```

```
Popped element is5
Enter your choice3
Elements in the stack are:2  4
Enter your choice4
exit
*/
```

2. Stacks using Linked List

```
struct Node
{
  int data;
  struct Node *next;
};
struct Node* top = NULL;
void push(int val) {
  struct Node* newnode = (struct Node*)malloc(sizeof(struct Node));
  newnode->data = val;
  newnode->next = top;
  top = newnode;
}
void pop() {
  if(top==NULL)
  cout<<"Stack Underflow"<<endl;
  else {
    cout<<"The popped element is "<< top->data <<endl;
    top = top->next;
  }
}
void display() {
  struct Node* ptr;
  if(top==NULL)
```

```
   cout<<"stack is empty";
   else {
      ptr = top;
      cout<<"Stack elements are: ";
      while (ptr != NULL) {
         cout<< ptr->data <<" ";
         ptr = ptr->next;
      }
   }
   cout<<endl;
}
int main() {
   int ch, val;
   cout<<"1) Push in stack"<<endl;
   cout<<"2) Pop from stack"<<endl;
   cout<<"3) Display stack"<<endl;
   cout<<"4) Exit"<<endl;
   do {
      cout<<"Enter choice: "<<endl;
      cin>>ch;
      switch(ch) {
         case 1: {
            cout<<"Enter value to be pushed:"<<endl;
            cin>>val;
            push(val);
            break;
         }
         case 2: {
            pop();
```

```
            break;
        }
        case 3: {
            display();
            break;
        }
        case 4: {
            cout<<"Exit"<<endl;
            break;
        }
        default: {
            cout<<"Invalid Choice"<<endl;
        }
      }
   }while(ch!=4);
   return 0;
}
/* Output :-
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit
Enter choice:
1
Enter value to be pushed:
2
Enter choice:
1
Enter value to be pushed:
```

4

Enter choice:

1

Enter value to be pushed:

6

Enter choice:

3

Stack elements are: 6 4 2

Enter choice:

2

The popped element is 6

Enter choice:

3

Stack elements are: 4 2

Enter choice:

4

Exit

*/

3. Evaluation of postfix expression

```
#include<iostream>
using namespace std;
class Evaluation {
        public:
                int st[50];
                int top;
                char str[50];
                Evaluation() {
                        top = -1;
```

```
        }
        void push(int val) {
                top++;
                st[top] = val;
        }
        int pop() {
                int val = st[top];
                top--;
                return val;
        }
        int operation(int a,int b,char opr) {
                switch(opr) {
                        case '+':return a+b;
                        case '-':return a-b;
                        case '*':return a*b;
                        case '/':return a/b;
                        default: return 0;
                }
        }
        int calculatePostfix();
};
int Evaluation::calculatePostfix() {
        int index = 0;
        while(str[index]!='\0') {
                if(isdigit(str[index])) {
                        push(str[index]-'0');
                }
                else {
                        int x = pop();
```

```
                int y = pop();
                int result = operation(y,x,str[index]);
                push(result);
            }
            index++;
        }
        return pop();
}
int main() {
        Evaluation eval;
        cout << "Enter the postfix: ";
        cin >> eval.str;
        int result = eval.calculatePostfix();
        cout << "The result is " << result;
        return 0;
}
/* Output :-
Enter the postfix: 231*+9-
The result is -4 */
```

4. Balancing Paranthesis

```
#include <bits/stdc++.h>
#include<stack>
#include<string>
#include <math.h>
using namespace std;


bool areBracketsBalanced(string expr)
{

   stack<char> temp;
   for (int i = 0; i < expr.length(); i++) {
      if (temp.empty()) {
```

```
                temp.push(expr[i]);
        }
        else if ((temp.top() == '(' && expr[i] == ')')
                || (temp.top() == '{' && expr[i] == '}')
                || (temp.top() == '[' && expr[i] == ']')) {

            temp.pop();
        }
        else {
            temp.push(expr[i]);
        }
    }
    if (temp.empty()) {


        return true;
    }
    return false;
}


int main()
{
    string expr;
     cout << "\n Enter a element";
     cin >> expr;

    if (areBracketsBalanced(expr))
        cout << "Balanced";
    else
        cout << "Not Balanced";
    return 0;
}


/* Output:
Enter a element  [{()}]
Balanced
*/
```