NAME: MOAZZAM FAROOQUI
TASK: DSA ASSIGNMENT#02
INSTRUCTOR: SAMIA MASOOD AWAN
COURSE CODE: CT-159
ROLL NO: CT-24068

Q1. Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid. An input string is valid if open brackets are closed by the same type of brackets in the correct order.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

class Node{
public:
    char data;
    Node*next;
    Node(char val){
        data=val;
        next=nullptr;
    }
};

class Stack{
private:
    Node*top;
public:
    Stack(){
        top=nullptr;
    }
    void push(char val){
        Node* newNode=new Node(val);
        newNode->next=top;
        top=newNode;
    }

    void pop(){
        if(top==NULL)return;
        Node* temp=top;
        top=top->next;
        delete temp;
    }

    char peek(){
        if(top==NULL)return -1;
        return top->data;
    }

    bool empty(){
        return top==NULL;
    }
```

```
    bool isValid(string s){
        Stack st;
        for(int i=0;i<s.length();i++){
            char ch=s[i];
            if(ch=='('||ch=='{'||ch=='[')st.push(ch);
            else{
                if(st.empty())return false;
                char top=st.peek();
                st.pop();
                if((ch==')'&&top!='(') || (ch=='}'&&top!='{') || (ch==']'&&top!='['))return false;
            }
        }
        return st.empty();
    }
};

int main(void){
    Stack st;
    string s;
    cout<<"ENTER STRING:";
    cin>>s;
    if(st.isValid(s))cout<<"VALID"<<endl;
    else cout<<"INVALID"<<endl;
    return 0;
}
```

OUTPUT:

```
ENTER STRING:{[()]}
VALID

--------------------------------
Process exited after 2.352 seconds with return value 0
Press any key to continue . . .
```

Q2. Use a stack to reverse a given string.

SOURCE CODE:

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

class Node{
public:
    char data;
    Node*next;
    Node(char val){
        data=val;
        next=nullptr;
    }
};

class Stack{
private:
    Node*top;
public:
    Stack(){
        top=nullptr;
    }
    void push(char val){
        Node*newnode=new Node(val);
        newnode->next=top;
        top=newnode;
    }
```

```
27      void pop(){
28          if(top==nullptr)return;
29          Node*temp=top;
30          top=top->next;
31          delete temp;
32      }
33
34      char peek(){
35          if(top==nullptr) return -1;
36          return top->data;
37      }
38
39      bool empty(){
40          return top==nullptr;
41      }
42  };
```

```
44  int main(void){
45      Stack st;
46      string s;
47      cout<<"ENTER STRING:";
48      cin>>s;
49
50      for(int i=0;i<s.length();i++)st.push(s[i]);
51      string reversed="";
52      while(!st.empty()){
53          reversed+=st.peek();
54          st.pop();
55      }
56
57      cout<<"REVERSED STRING:"<<reversed<<endl;
58      return 0;
59  }
```

OUTPUT:

```
ENTER STRING:MOAZZAM
REVERSED STRING:MAZZAOM

--------------------------------
Process exited after 13.33 seconds with return value 0
Press any key to continue . . .
```

Q3. Write a program that takes a decimal number and uses a stack to convert it to its binary representation.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

class Node{
public:
    int data;
    Node*next;
    Node(int val){
        data=val;
        next=nullptr;
    }
};

class Stack{
private:
    Node*top;
public:
    Stack(){
        top=nullptr;
    }

    void push(int val){
        Node*newnode=new Node(val);
        newnode->next=top;
        top=newnode;
    }
    void pop(){
        if(top==nullptr)return;
        Node*temp=top;
        top=top->next;
        delete temp;
    }

    int peek(){
        if(top==nullptr)return -1;
        return top->data;
    }

    bool empty(){
        return top==nullptr;
    }
};
```

```
int main(void){
    int num;
    cout<<"ENTER A DECIMAL NUMBER: ";
    cin>>num;

    Stack st;
    int n=num;
    while(n>0){
        int rem=n%2;
        st.push(rem);
        n=n/2;
    }
    cout<<"BINARY REPRESENTATION OF "<<num<<" IS: ";
    while(!st.empty()){
        cout<<st.peek();
        st.pop();
    }
    cout<<endl;
    return 0;
}
```

OUTPUT:

```
ENTER A DECIMAL NUMBER: 12
BINARY REPRESENTATION OF 12 IS: 1100

--------------------------------
Process exited after 3.845 seconds with return value 0
Press any key to continue . . .
```

Q4. Implement an algorithm that converts an arithmetic expression from infix notation to postfix notation using a stack. For example, (A + B) * C becomes AB+C*.

SOURCE CODE:

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

class Node{
public:
    char data;
    Node*next;
    Node(char val){
        data=val;
        next=nullptr;
    }
};

class Stack{
    Node*top;
public:
    Stack(){
        top=nullptr;
    }

    void push(char c){
        Node*newnode=new Node(c);
        newnode->next=top;
        top=newnode;
    }
```

```
    char pop(){
        if(top==nullptr)return -1;
        Node*temp=top;
        char val=temp->data;
        top=top->next;
        delete temp;
        return val;
    }

    char peek(){
        if(top==nullptr)return -1;
        return top->data;
    }

    bool empty(){
        return top==nullptr;
    }
};
```

```
int precedence(char op){
    if(op=='^')return 3;
    if(op=='*'||op=='/')return 2;
    if(op=='+'||op=='-')return 1;
    return 0;
}

bool isoperand(char c){
    return c>='A'&&c<='Z';
}
```

```
string infixtopostfix(string infix){
    Stack st;
    string postfix="";
    for(int i=0;i<infix.size();i++){
        char c=infix[i];
        if(isoperand(c))postfix+=c;
        else if(c=='(')st.push(c);
        else if(c==')'){
            while(!st.empty()&&st.peek()!='(')postfix+=st.pop();
            st.pop();
        }
        else{
            while(!st.empty()&&precedence(st.peek())>=precedence(c))postfix+=st.pop();
            st.push(c);
        }
    }
    while(!st.empty())postfix+=st.pop();
    return postfix;
}

int main(void){
    string infix="(A+B)*C";
    cout<<"INFIX:"<<infix<<endl;
    cout<<"POSTFIX:"<<infixtopostfix(infix)<<endl;
    return 0;
}
```

OUTPUT:

```
INFIX:(A+B)*C
POSTFIX:AB+C*

--------------------------------
Process exited after 0.438 seconds with return value 0
Press any key to continue . . .
```

Q5. Write a function that evaluates the postfix expression P = 5 6 2 + * 12 4 / -

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

class Node{
public:
    int data;
    Node*next;
    Node(int val){
        data=val;
        next=nullptr;
    }
};

class Stack{
    Node*top;
public:
    Stack(){
        top=nullptr;
    }

    void push(int val){
        Node*newnode=new Node(val);
        newnode->next=top;
        top=newnode;
    }
    int pop(){
        if(top==nullptr){
            cout<<"STACK UNDERFLOW!"<<endl;
            return -1;
        }
        int val=top->data;
        Node*temp=top;
        top=top->next;
        delete temp;
        return val;
    }

    bool isempty(){
        return top==nullptr;
    }
};
```

```
int evaluatepostfix(string exp[],int n){
    Stack st;
    for(int i=0;i<n;i++){
        string token=exp[i];
        if(token=="+" || token=="-" || token=="*" || token=="/"){
            int b=st.pop();
            int a=st.pop();
            int result;
            if(token=="+")result=a+b;
            else if(token=="-")result=a-b;
            else if(token=="*")result=a*b;
            else result=a/b;
            st.push(result);
        }
        else{
            int num=stoi(token);
            st.push(num);
        }
    }
    return st.pop();
}
```

```
int main(void){
    string p[]={"5","6","2","+","*","12","4","/","-"};
    int size=sizeof(p)/sizeof(p[0]);
    cout<<"RESULT:"<<evaluatepostfix(p,size)<<endl;
    return 0;
}
```

OUTPUT:

```
RESULT:37

--------------------------------
Process exited after 0.3925 seconds with return value 0
Press any key to continue . . .
```

Q6. Design a stack that, in addition to the standard push and pop operations, also has a get_min operation that returns the minimum element. All three operations should have a time complexity of O(1).

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

class Node{
public:
    int data;
    Node*next;
    Node(int val){
        data=val;
        next=nullptr;
    }
};

class Stack{
    Node*top;
    Node*mintop;
public:
    Stack(){
        top=nullptr;
        mintop=nullptr;
    }
    void push(int val){
        Node*newnode=new Node(val);
        newnode->next=top;
        top=newnode;
        if(mintop==nullptr || val<=mintop->data){
            Node*newmin=new Node(val);
            newmin->next=mintop;
            mintop=newmin;
        }
    }
    int pop(){
        if(top==nullptr){
            cout<<"STACK UNDERFLOW!"<<endl;
            return -1;
        }
        int val=top->data;
        if(val==mintop->data){
            Node*tempmin=mintop;
            mintop=mintop->next;
            delete tempmin;
        }
        Node*temp=top;
        top=top->next;
        delete temp;
        return val;
    }
```

```
51      int peek(){
52          if(top==nullptr){
53              cout<<"STACK IS EMPTY!"<<endl;
54              return -1;
55          }
56          return top->data;
57      }
58
59      int getmin(){
60          if(mintop==nullptr){
61              cout<<"STACK IS EMPTY!"<<endl;
62              return -1;
63          }
64          return mintop->data;
65      }
66
67      bool isempty(){
68          return top==nullptr;
69      }
70  };
```

```
72  int main(void){
73      Stack st;
74      st.push(5);
75      st.push(3);
76      st.push(7);
77      st.push(2);
78      st.push(6);
79      cout<<"CURRENT MIN:"<<st.getmin()<<endl;
80      st.pop();
81      st.pop();
82      cout<<"CURRENT MIN:"<<st.getmin()<<endl;
83      return 0;
84  }
```

OUTPUT:

```
CURRENT MIN:2
CURRENT MIN:3

--------------------------------
Process exited after 0.4053 seconds with return value 0
Press any key to continue . . .
```

Q7.Given an array of integers representing the heights of a histogram's bars, where each bar has a width of 1, find the area of the largest rectangle in the histogram. Use a stack-based approach to solve this in O(n) time.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

class Node{
public:
    int index;
    int height;
    Node*next;
    Node(int i,int h){
        index=i;
        height=h;
        next=nullptr;
    }
};

class Stack{
    Node*top;
public:
    Stack(){
        top=nullptr;
    }

    void push(int idx,int h){
        Node*newnode=new Node(idx,h);
        newnode->next=top;
        top=newnode;
    }
```

```cpp
    void pop(){
        if(top==nullptr)return;
        Node*temp=top;
        top=top->next;
        delete temp;
    }

    bool empty(){
        return top==nullptr;
    }

    int peekindex(){
        if(top==nullptr)return -1;
        return top->index;
    }

    int peekheight(){
        if(top==nullptr)return -1;
        return top->height;
    }
};
```

```
int largestrectanglearea(int heights[],int n){
    Stack st;
    int maxarea=0;
    for(int i=0;i<=n;i++){
        int h;
        if(i==n)h=0;
        else h=heights[i];
        int lastindex=i;
        while(!st.empty()&&h<st.peekheight()){
            int height=st.peekheight();
            int index=st.peekindex();
            st.pop();
            lastindex=index;
            int width=i-index;
            int area=height*width;
            if(area>maxarea)maxarea=area;
        }
        st.push(lastindex,h);
    }
    return maxarea;
}
```

```
int main(void){
    int heights[]={2,1,5,6,2,3};
    int n=sizeof(heights)/sizeof(heights[0]);
    cout<<"LARGEST RECTANGLE AREA:"<<largestrectanglearea(heights,n)<<endl;
    return 0;
}
```

OUTPUT:

```
LARGEST RECTANGLE AREA:10

--------------------------------
Process exited after 0.403 seconds with return value 0
Press any key to continue . . .
```

Q8. Implement a queue using a list or an array. Include the basic operations: enqueue (add an element), dequeue (remove an element), peek (view the front element), and is_empty.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

class Node{
public:
    int data;
    Node*next;
    Node(int val){
        data=val;
        next=nullptr;
    }
};

class Queue{
    Node*front;
    Node*rear;
public:
    Queue(){
        front=nullptr;
        rear=nullptr;
    }
    void enqueue(int x){
        Node*newnode=new Node(x);
        if(rear==nullptr){
            front=rear=newnode;
            cout<<"ENQUEUED:"<<x<<endl;
            return;
        }
        rear->next=newnode;
        rear=newnode;
        cout<<"ENQUEUED:"<<x<<endl;
    }

    int dequeue(){
        if(isempty()){
            cout<<"QUEUE IS EMPTY!"<<endl;
            return -1;
        }
        Node*temp=front;
        int val=temp->data;
        front=front->next;
        if(front==nullptr)rear=nullptr;
        delete temp;
        cout<<"DEQUEUED:"<<val<<endl;
        return val;
    }
```

```
49      int peek(){
50          if(isempty()){
51              cout<<"QUEUE IS EMPTY!"<<endl;
52              return -1;
53          }
54          return front->data;
55      }
56
57      bool isempty(){
58          return front==nullptr;
59      }
60  };
61
62  int main(void){
63      Queue q;
64      q.enqueue(10);
65      q.enqueue(20);
66      q.enqueue(30);
67      cout<<"FRONT:"<<q.peek()<<endl;
68      q.dequeue();
69      cout<<"FRONT:"<<q.peek()<<endl;
70      return 0;
71  }
```

OUTPUT:

```
ENQUEUED:10
ENQUEUED:20
ENQUEUED:30
FRONT:10
DEQUEUED:10
FRONT:20

--------------------------------
Process exited after 0.4894 seconds with return value 0
Press any key to continue . . .
```

Q9. Implement a queue's enqueue and dequeue operations using only two stacks.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

class Node{
public:
    int data;
    Node*next;
    Node(int val){
        data=val;
        next=nullptr;
    }
};

class Stack{
    Node*top;
public:
    Stack(){
        top=nullptr;
    }

    void push(int val){
        Node*newnode=new Node(val);
        newnode->next=top;
        top=newnode;
    }
    int pop(){
        if(top==nullptr) return -1;
        Node*temp=top;
        int val=temp->data;
        top=top->next;
        delete temp;
        return val;
    }

    bool empty(){
        return top==nullptr;
    }

    int peek(){
        if(top==nullptr) return -1;
        return top->data;
    }
};
```

```
46  class Queue{
47  private:
48      Stack s1,s2;
49  public:
50      void enqueue(int x){
51          s1.push(x);
52          cout<<"ENQUEUED:"<<x<<endl;
53      }
54
55      int dequeue(){
56          if(s2.empty()){
57              while(!s1.empty()){
58                  s2.push(s1.pop());
59              }
60          }
61          if(s2.empty()){
62              cout<<"QUEUE IS EMPTY!"<<endl;
63              return -1;
64          }
65          int val=s2.pop();
66          cout<<"DEQUEUED:"<<val<<endl;
67          return val;
68      }
69  };
```

```
71  int main(void){
72      Queue q;
73      q.enqueue(10);
74      q.enqueue(20);
75      q.enqueue(30);
76      q.dequeue();
77      q.dequeue();
78      q.enqueue(40);
79      q.dequeue();
80      q.dequeue();
81      q.dequeue();
82      return 0;
83  }
```

OUTPUT:

```
ENQUEUED:10
ENQUEUED:20
ENQUEUED:30
DEQUEUED:10
DEQUEUED:20
ENQUEUED:40
DEQUEUED:30
DEQUEUED:40
QUEUE IS EMPTY!

--------------------------------
Process exited after 0.03379 seconds with return value 0
Press any key to continue . . .
```

Q10. Given an array and a window size k, find the maximum element in each sliding window of size k. Use a deque (double-ended queue) to solve this problem efficiently in O(n) time.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

vector<int>slidingwindowmax(vector<int>&arr,int k){
    deque<int>dq;
    vector<int>result;
    for(int i=0;i<arr.size();i++){
        while(!dq.empty()&&dq.front()<=i-k)dq.pop_front();
        while(!dq.empty()&&arr[dq.back()]<=arr[i])dq.pop_back();
        dq.push_back(i);
        if(i>=k-1)result.push_back(arr[dq.front()]);
    }
    return result;
}

int main(void){
    vector<int>arr={1,3,-1,-3,5,3,6,7};
    int k=3;
    vector<int>ans=slidingwindowmax(arr,k);
    cout<<"SLIDING WINDOW MAXIMUMS:";
    for(int i=0;i<ans.size();i++)cout<<" "<<ans[i];
    cout<<endl;
    return 0;
}
```

OUTPUT:

```
SLIDING WINDOW MAXIMUMS: 3 3 5 5 6 7

--------------------------------
Process exited after 0.3665 seconds with return value 0
Press any key to continue . . .
```

Q11. You are given a queue with N elements. You can perform two operations: Left-Shift (dequeue from the front) and Right-Shift (dequeue from the rear). Implement an algorithm to find the minimum number of operations to make the queue empty.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

class Node{
public:
    int data;
    Node*next;
    Node*prev;
    Node(int val){
        data=val;
        next=nullptr;
        prev=nullptr;
    }
};

class Deque{
    Node*front;
    Node*rear;
    int size;
public:
    Deque(){
        front=nullptr;
        rear=nullptr;
        size=0;
    }
```

```cpp
    void enqueue(int val){
        Node*newnode=new Node(val);
        if(rear==nullptr){
            front=newnode;
            rear=newnode;
        }
        else{
            rear->next=newnode;
            newnode->prev=rear;
            rear=newnode;
        }
        size++;
    }
```

```
    void leftshift(){
        if(front==nullptr){
            cout<<"QUEUE UNDERFLOW(FRONT)"<<endl;
            return;
        }
        cout<<front->data<<" REMOVED FROM FRONT"<<endl;
        Node*temp=front;
        front=front->next;
        if(front)front->prev=nullptr;
        else rear=nullptr;
        delete temp;
        size--;
    }
    void rightshift(){
        if(rear==nullptr){
            cout<<"QUEUE UNDERFLOW(REAR)"<<endl;
            return;
        }
        cout<<rear->data<<" REMOVED FROM REAR"<<endl;
        Node*temp=rear;
        rear=rear->prev;
        if(rear)rear->next=nullptr;
        else front=nullptr;
        delete temp;
        size--;
    }
```

```
        int getsize(){
            return size;
        }

        void display(){
            Node*curr=front;
            cout<<"QUEUE:";
            while(curr){
                cout<<" "<<curr->data;
                curr=curr->next;
            }
            cout<<endl;
        }
};
```

```
int main(void){
    int n;
    cout<<"ENTER SIZE OF QUEUE:";
    cin>>n;

    Deque q;
    for(int i=1;i<=n;i++)q.enqueue(i);
    q.display();
    cout<<"MINIMUM OPERATIONS NEEDED="<<q.getsize()<<endl;
    cout<<"PERFORMING DELETIONS:"<<endl;
    while(q.getsize()>0){
        if(q.getsize()%2==0)q.leftshift();
        else q.rightshift();
    }
    return 0;
}
```

OUTPUT:

```
ENTER SIZE OF QUEUE:3
QUEUE: 1 2 3
MINIMUM OPERATIONS NEEDED=3
PERFORMING DELETIONS:
3 REMOVED FROM REAR
1 REMOVED FROM FRONT
2 REMOVED FROM REAR

--------------------------------
Process exited after 4.039 seconds with return value 0
Press any key to continue . . .
```