DSA Lab06

23K2001

M.Muzammil Siddiqui

BCS-3J

Q1:

```cpp
//23K2001 - Muzammil
#include<iostream>
using namespace std;
class stacks{
    private:
        int top,size;
    public:
        char *arr;
        stacks():top(-1),size(0),arr(nullptr){}
        stacks(int s):top(-1),size(s){
            arr = new char[size];
            for(int i=0;i<size;i++)
                arr[i]='!';
        }

        void push(char e){
            if(top>=(size-1)){
                cout<<"Stack overflow occured!"<<endl;
                return;
            }
            arr[++top] = e;
        }
        char pop(){
            if(top<0){
                cout<<"Stacks underflow occured!"<<endl;
                return '!';
            }
            char last = arr[top--];
            return last;
        }

        bool checkPalindrome(){
            stacks reversed(this->size);
            for(int i=0;i<size;i++)
                reversed.push(this->arr[i]);

            for(int i=0;i<size;i++){
                if(this->arr[i]!=reversed.pop())
                    return false;
            }
            return true;
        }
```

```
        ~stacks(){ delete[] arr; }
};
int main(){
    int s;
    cout<<"Enter size of stack: ";
    cin>>s;
    stacks flex(s);
    char e;
    cout<<"Enter "<<s<<" elements:"<<endl;
    for(int i=0;i<s;i++){
        cin>>e;
        flex.push(e);
    }
    cout<<endl<<"Checking for Palindrome:"<<endl;
    if(flex.checkPalindrome())
        cout<<"Yes!"<<endl;
    else
        cout<<"No!"<<endl;
    return 0;
}
```

```
Enter size of stack: 9
Enter 9 elements:
B
O
R
R
O
W
R
O
B

Checking for Palindrome:
No!
PS F:\Semester Material -
```

```
Enter size of stack: 7
Enter 7 elements:
R
A
C
E
C
A
R

Checking for Palindrome:
Yes!
PS F:\Semester Material -
```

Q2:

```cpp
//23K2001 - Muzammil
#include<iostream>
using namespace std;
class stacks{
    private:
        int top,size;
    public:
        string *arr;
        stacks():top(-1),size(0),arr(nullptr){}
        stacks(int s):top(-1),size(s){
            arr = new string[size];
            for(int i=0;i<size;i++)
                arr[i]="!";
        }

        void push(string e){
            if(top>=(size-1)){
                cout<<"Stack overflow occured!"<<endl;
                return;
            }
            arr[++top] = e;
        }
        string pop(){
            if(top<0){
                cout<<"Stacks underflow occured!"<<endl;
                return "!";
            }
            string last = arr[top--];
            return last;
        }

        bool checkPalindrome(){
            stacks reversed(this->size);
            for(int i=0;i<size;i++)
                reversed.push(this->arr[i]);

            for(int i=0;i<size;i++){
                if(this->arr[i]!=reversed.pop())
                    return false;
            }
            return true;
        }
```

```
        bool isEmpty(){
            if(top<0)
                return true;
            return false;
        }
        ~stacks(){ delete[] arr; }
};
int main(){
    int s;
    cout<<"Enter to-do list size: ";
    cin>>s;
    stacks flex(s);
    string e;
    cout<<"Enter "<<s-1<<" tasks:"<<endl;
    for(int i=0;i<s-1;i++){
        cin>>e;
        flex.push(e);
    }
    cout<<endl<<"Enter a task to add to top:"<<endl;
    cin>>e;
    flex.push(e);
    cout<<endl<<"Checking if stack is empty:"<<endl;
    if(flex.isEmpty())
        cout<<"Yes!"<<endl;
    else
        cout<<"No!"<<endl;

    cout<<endl<<"Removing top task from the list."<<endl;
    e = flex.pop();
    cout<<"Last task was: "<<e<<endl;
    return 0;
}
```

```
Enter to-do list size: 4
Enter 3 tasks:
Check-in
Sign
Meeting

Enter a task to add to top:
Lunch

Checking if stack is empty:
No!

Removing top task from the list.
Last task was: Lunch
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3
```

Q3:

```
//23K2001 - Muzammil
#include<iostream>
using namespace std;
class node{
    private:
        string data;
        node* next;
    public:
        node() : next(nullptr){}
        node(string s) : next(nullptr), data(s){}
        string getData(){ return data; }
        void setData(string s){ data=s; }
        node* getNext(){ return next; }
        void setNext(node* n){ next=n; }
};
class stacks{
    private:
        node* top;
    public:
        stacks():top(nullptr){}
        void push(string e){
            node* n = new node;
            if(n==nullptr){
                cout<<"Stack overflow occured!"<<endl;
                return;
            }
            n->setData(e);
            n->setNext(top);
            top = n;
        }
        string pop(){
            if(top==nullptr){
                cout<<"Stacks underflow occured!"<<endl;
                return "!";
            }
            string last = top->getData();
            node* temp = top;
            top=top->getNext();
            delete temp;
            return last;
        }
        string peek(){
```

```
            if(isEmpty())
                return "!";
            string last = top->getData();
            return last;
        }
        void display(){
            if(top==nullptr){
                cout<<"Stack empty!"<<endl;
                return;
            }
            node* temp = top;
            while(temp!=nullptr){
                cout<<temp->getData()<<endl;
                temp=temp->getNext();
            }

        }
        bool isEmpty(){
            if(top==nullptr)
                return true;
            return false;
        }
};
int main(){
    stacks flex;
    flex.push("Google");
    flex.push("Facebook");
    flex.push("Twitter");
    flex.push("LinkedIn");
    flex.push("Instagram");

    cout<<"Stack list:"<<endl;
    flex.display();
    cout<<endl<<"Popped two sites:"<<endl;
    cout<<flex.pop()<<endl;
    cout<<flex.pop()<<endl;
    cout<<"Top: "<<flex.peek()<<endl;

    cout<<endl<<"Stack list:"<<endl;
    flex.display();
    return 0;
}
```

```
Stack list:
Instagram
LinkedIn
Twitter
Facebook
Google

Popped two sites:
Instagram
LinkedIn
Top: Twitter

Stack list:
Twitter
Facebook
Google
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures
```

Q4:

```
//23K2001 - Muzammil
#include<iostream>
using namespace std;
class stacks{
    private:
        int top,size;
    public:
        char *arr;
        stacks():top(-1),size(0),arr(nullptr){}
        stacks(int s):top(-1),size(s){
            arr = new char[size];
            for(int i=0;i<size;i++)
                arr[i]='!';
        }
        int getpeek(){ return top; }
        int getSize() { return size; }

        void push(char e){
            if(top>=(size-1)){
                cout<<"Stack overflow occured!"<<endl;
                return;
            }
            arr[++top] = e;
        }
        char pop(){
            if(top<0){
                cout<<"Stacks underflow occured!"<<endl;
                return '!';
            }
            char last = arr[top--];
            return last;
        }
        char peek(){
            if(top < 0){
                cout << "Stack is Empty";
                return 0;
            } else{
                int x = arr[top];
                return x;
            }
        }
        bool isEmpty(){ return (top<0); }
```

```
};
int precedence(char c){
    if(c == '^')
        return 3;
    else if(c == '*' || c == '/')
        return 2;
    else if(c == '+' || c == '-')
        return 1;
    else
        return -1;
}
string infixToPostfix(string infix){
    string postfix = "";
    stacks s(infix.length());

    for(int i = 0; i < infix.length(); i++){
        char c = infix[i];

        if((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z'))
            postfix += c;
        else if(c == '(')
            s.push(c);
        else if(c == ')'){
            while(!s.isEmpty() && s.peek() != '('){
                char op = s.pop();
                postfix += op;
            }
            if(s.peek() == '(')
                s.pop();
        } else{
            while(!s.isEmpty() && precedence(c) <= precedence(s.peek())){
                char op = s.pop();
                postfix += op;
            }
            s.push(c);
        }
    }
    while(!s.isEmpty()){
        char op = s.pop();
        postfix += op;
    }

    return postfix;
}
int main(){
```

```
    cout<<"\t\tWelcome to Infix to Postfix expression converter!"<<endl;
    cout<<endl<<"Enter you infix expression: "<<endl;
    string e;
    cin>>e;
    cout<<endl<<"After conversion :-"<<endl;
    cout<<"Postfix: "<<infixToPostfix(e)<<endl;
    // a+b*(c^d-e)^(f+g*h)-i
    return 0;
}
```

```
                Welcome to Infix to Postfix expression converter!

Enter you infix expression:
a+b*(c^d-e)^(f+g*h)-i

After conversion :-
Postfix: abcd^e-fgh*+^*+i-
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)\Lab Tasks\
```

Q5:

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

template<class T>
class node{
private:
    T data;
    node<T>* next;
public:
    node() : next(nullptr){}
    node(T s) : next(nullptr), data(s){}
    T getData(){ return data; }
    void setData(T s){ data = s; }
    node<T>* getNext(){ return next; }
    void setNext(node<T>* n){ next = n; }
};
template<class T>
class stack{
private:
    node<T>* top;
public:
    stack() : top(nullptr){}

    void push(T e){
        node<T>* n = new node<T>;
        if(n == nullptr){
            cout<<"Stack overflow occurred!"<<endl;
            return;
        }
        n->setData(e);
        n->setNext(top);
        top = n;
    }

    T pop(){
        if(top == nullptr){
            cout<<"Stack underflow occurred!"<<endl;
            return T();
        }
        T last = top->getData();
```

```
        node<T>* temp = top;
        top = top->getNext();
        delete temp;
        return last;
    }

    T peek(){
        if(isEmpty())
            return T();
        return top->getData();
    }

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

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

float applyOperation(char op,float val1,float val2){
    switch(op){
        case '+': return val1 + val2;
        case '-': return val1 - val2;
        case '*': return val1 * val2;
        case '/': return val1 / val2;
        case '^': return pow(val1, val2);
        default: return 0;
    }
}

bool checkDigit(char c){ return (c >= '0' && c <= '9'); }

float parseNumber(const string& s, int& i){
    string numStr = "";
    while(i < s.length() && (checkDigit(s[i]) || s[i] == '.')){
        numStr += s[i];
        i++;
    }
```

```
    i--;
    return stof(numStr);
}

void evaluateTop(stack<char>& operators, stack<float>& values){
    if(operators.isEmpty()) return;

    char op = operators.pop();
    float val2 = values.pop();
    float val1 = values.pop();

    values.push(applyOperation(op,val1,val2));
}

float calcExpression(const string& infix){
    stack<char> operators;
    stack<float> values;

    for(int i = 0; i < infix.length(); i++){
        char c = infix[i];

        if(checkDigit(c) || c == '.')
            values.push(parseNumber(infix,i));
        else if(c == '('){
            if(i > 0 && (checkDigit(infix[i-1]) || infix[i-1] == ')'))
                operators.push('*');

            operators.push(c);
        }
        else if(c == ')'){
            while (!operators.isEmpty() && operators.peek() != '(')
                evaluateTop(operators,values);

            operators.pop();
        }
        else if(c == '*' || c == '/' || c == '+' || c == '-' || c == '^'){
            while (!operators.isEmpty() && precedence(c) <=
precedence(operators.peek()))
                evaluateTop(operators,values);

            operators.push(c);
        }
    }
    while (!operators.isEmpty())
        evaluateTop(operators, values);
```

```
    return values.peek();
}

int main() {
    cout<<"\t\tWelcome to Muzammil's Calculator!"<<endl;
    cout<<endl<<"Enter your expression: "<<endl;
    string e;
    cin>>e;

    float result = calcExpression(e);
    cout<<"Result: "<<result<<endl;
    return 0;
}
```

```
                Welcome to Muzammil's Calculator!

Enter your expression:
12+13-5(0.5+0.5)+1
Result: 21
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)
er-3\Data Structures (LAB)\Lab Tasks\Lab06 - Stacks & Queues\" ; if ($?) { g
                Welcome to Muzammil's Calculator!

Enter your expression:
12+13-5(1)+1
Result: 21
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)
er-3\Data Structures (LAB)\Lab Tasks\Lab06 - Stacks & Queues\" ; if ($?) { g
                Welcome to Muzammil's Calculator!

Enter your expression:
12+13-5*(0.5+0.5)+1
Result: 21
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)
```

Q6:

```
//23K2001 - Muzammil
#include<iostream>
using namespace std;
class order{
    private:
    int qty;
    string item;
    public:
    order(){}
    order(string i, int q):qty(q),item(i){}
    void setQty(int q){ qty = q; }
    void setItem(string i){ item = i; }
    string getItem(){ return item; }
    int getQty(){ return qty; }
    void setOrder(string i, int q){
        setQty(q);
        setItem(i);
    }
    void getOrder(){ cout<<"Item: "<<getItem()<<" - Quantity: "<<getQty()<<endl;
}
};
class queue{
    private:
        int front,rear,size;
    public:
        order *arr;
        queue():front(-1),rear(-1),size(0),arr(nullptr){}
        queue(int s):front(-1),rear(-1),size(s){
            arr = new order[size];
            for(int i=0;i<size;i++)
                arr[i].setOrder("",0);
        }
        void enqueue(string i,int q){
            if(isFull()){
                cout<<"Queue is full!"<<endl;
                return;
            }
            else if(isEmpty())
                front = rear = 0;
            else
                rear++;
            arr[rear].setOrder(i,q);
```

```
        }
        void dequeue(){
            if(isEmpty()){
                cout<<"Queue is empty!"<<endl;
                return;
            }
            else if(front == rear){
                arr[front].getOrder();
                front = rear = -1;
            }
            else{
                arr[front].getOrder();
                front++;
            }
        }
        bool isEmpty(){
            if(front==-1 || front>rear)
                return true;
            return false;
        }
        bool isFull(){
            if(rear==size-1)
                return true;
            return false;
        }
        void display(){
            if(isEmpty()){
                cout<<"Queue is empty!"<<endl;
                return;
            }
            cout<<endl<<"Orders in the queue are: "<<endl;
            for(int i=front;i<=rear;i++){
                cout<<i<<". ";
                arr[i].getOrder();
            }
        }
        ~queue(){ delete[] arr; }
};
int main(){
    int s;
    cout<<"Enter number of orders: ";
    cin>>s;
    queue orders(s);
    string o;
    int q;
```

```
    cout<<"Place "<<s<<" orders: [Item] [Quantity]"<<endl;
    for(int i=0;i<s;i++){
        cin>>o>>q;
        orders.enqueue(o,q);
    }
    orders.display();
    cout<<endl<<"Processing orders.."<<endl;
    while(!orders.isEmpty())
        orders.dequeue();

    if(orders.isEmpty())
        cout<<endl<<"All orders have been processed!"<<endl;
    else
        cout<<endl<<"All orders have NOT been processed yet!"<<endl;
    return 0;
}
```

```
Enter number of orders: 3
Place 3 orders: [Item] [Quantity]
nihari 2
haleem 4
biryani 8

Orders in the queue are:
0. Item: nihari - Quantity: 2
1. Item: haleem - Quantity: 4
2. Item: biryani - Quantity: 8

Processing orders..
Item: nihari - Quantity: 2
Item: haleem - Quantity: 4
Item: biryani - Quantity: 8

All orders have been processed!
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures
```

Q7:

```
//23K2001 - Muzammil
#include<iostream>
using namespace std;
class queue{
    private:
        int front,rear,size;
    public:
        int *ids;
        queue():front(-1),rear(-1),size(0),ids(nullptr){}
        queue(int s):front(-1),rear(-1),size(s){
            ids = new int[size];
            for(int i=0;i<size;i++)
                ids[i]=-1;
        }
        void enqueue(int q){
            if(isFull()){
                cout<<"Queue is full!"<<endl;
                return;
            }
            else if(isEmpty())
                front = rear = 0;
            else
                rear++;
            ids[rear]=q;
        }
        void dequeue(){
            if(isEmpty()){
                cout<<"Queue is empty!"<<endl;
                return;
            }
            else if(front == rear){
                cout<<"CustomerID#"<<ids[front]<<" processed!"<<endl;
                front = rear = -1;
            }
            else{
                cout<<"CustomerID#"<<ids[front]<<" processed!"<<endl;
                front++;
            }
        }
        bool isEmpty(){
            if(front==-1 || front>rear)
                return true;
```

```
            return false;
        }
        bool isFull(){
            if(rear==size-1)
                return true;
            return false;
        }
        void display(){
            if(isEmpty()){
                cout<<"Queue is empty!"<<endl;
                return;
            }
            cout<<endl<<"Customer IDs in the queue are: "<<endl;
            for(int i=front;i<=rear;i++)
                cout<<i<<". "<<ids[i]<<endl;
        }
        ~queue(){ delete[] ids; }
};
int main(){
    int s;
    cout<<"Enter number of customers: ";
    cin>>s;
    queue flex(s);
    int q;
    cout<<"Enter "<<s<<" IDs:"<<endl;
    for(int i=0;i<s;i++){
        cin>>q;
        flex.enqueue(q);
    }
    flex.display();
    cout<<endl<<"Making checkouts.."<<endl;
    while(!flex.isEmpty())
        flex.dequeue();

    if(flex.isEmpty())
        cout<<endl<<"All checkouts have been processed!"<<endl;
    else
        cout<<endl<<"All checkouts have NOT been processed yet!"<<endl;
    return 0;
}
```

```
Enter number of customers: 7
Enter 7 IDs:
13 7 4 1 6 8 10

Customer IDs in the queue are:
0. 13
1. 7
2. 4
3. 1
4. 6
5. 8
6. 10

Making checkouts..
CustomerID#13 processed!
CustomerID#7 processed!
CustomerID#4 processed!
CustomerID#1 processed!
CustomerID#6 processed!
CustomerID#8 processed!
CustomerID#10 processed!

All checkouts have been processed!
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures
```

Q8:

```
// 23K2001 - Muzammil
#include<iostream>
using namespace std;

class queue {
    private:
        int front, rear, size;
    public:
        string *arr;
        queue(): front(-1), rear(-1), size(0), arr(nullptr){}
        queue(int s): front(-1), rear(-1), size(s){
            arr = new string[size];
            for(int i = 0; i < size; i++)
                arr[i] = "!";
        }
        void enqueue(string message){
            if(isFull()){
                cout << "Message queue is full!" << endl;
                return;
            }
            else if(isEmpty())
                front = rear = 0;
            else
                rear++;
            arr[rear] = message;
            cout<<"Message '"<<message<<"' has been added to queue."<<endl;
        }
        void dequeue(){
            if(isEmpty()){
                cout<<"Message queue is empty!"<<endl;
                return;
            }
            else if(front == rear)
                front = rear = -1;
            else
                front++;
        }
        string atFront(){
            if(isEmpty()){
                cout<<"Message queue is empty!"<<endl;
                return "!";
            }
```

```
            return arr[front];
        }
        bool isEmpty(){ return front == -1 || front > rear; }
        bool isFull(){ return rear == size - 1; }
        void display(){
            if(isEmpty()){
                cout<<"Message queue is empty!"<<endl;
                return;
            }
            cout<<endl<<"Messages in queue: "<<endl;
            for(int i = front;i<=rear;i++)
                cout<<i<<". "<<arr[i]<<endl;
        }
        ~queue(){ delete[] arr; }
};

const int maxMessages = 15;

int main(){
    queue messages(maxMessages);
    int choice;
    string s;
    do{
        cout<<endl<<"\t\tFAST NU WHATSAPP LITE FYP"<<endl;
        cout<<"1. Add message to queue"<<endl;
        cout<<"2. Process message"<<endl;
        cout<<"3. Display messages in queue"<<endl;
        cout<<"4. Exit"<<endl;
        cout<<"Enter your choice: ";
        cin>>choice;

        switch(choice){
            case 1: {
                cout<<"Enter message: ";
                cin.ignore();
                getline(cin,s);
                messages.enqueue(s);
                break;
            }
            case 2: {
                if(!messages.isEmpty()){
                    cout<<"Processing message: "<<messages.atFront() << endl;
                    messages.dequeue();
                }
                else
```

```
                    cout<<"No messages in the queue."<<endl;
                break;
            }
            case 3: {
                if(messages.isEmpty())
                    cout<<"The message queue is empty."<<endl;
                else
                    messages.display();
                break;
            }
            case 4:
                break;
            default:
                cout<<"Invalid choice! Please try again."<<endl;
        }
    } while(choice != 4);

    return 0;
}
```

```
                FAST NU WHATSAPP LITE FYP
1. Add message to queue
2. Process message
3. Display messages in queue
4. Exit
Enter your choice: 1
Enter message: hello
Message 'hello' has been added to queue.

                FAST NU WHATSAPP LITE FYP
1. Add message to queue
2. Process message
3. Display messages in queue
4. Exit
Enter your choice: 1
Enter message: bye
Message 'bye' has been added to queue.

                FAST NU WHATSAPP LITE FYP
1. Add message to queue
2. Process message
3. Display messages in queue
4. Exit
Enter your choice: 3

Messages in queue:
0. hello
1. bye
```

```
                FAST NU WHATSAPP LITE FYP
1. Add message to queue
2. Process message
3. Display messages in queue
4. Exit
Enter your choice: 2
Processing message: hello

                FAST NU WHATSAPP LITE FYP
1. Add message to queue
2. Process message
3. Display messages in queue
4. Exit
Enter your choice: 3

Messages in queue:
1. bye

                FAST NU WHATSAPP LITE FYP
1. Add message to queue
2. Process message
3. Display messages in queue
4. Exit
Enter your choice: 4
PS F:\Semester Material - Muzammil\FAST-KHI-Seme
```

Q9:

```
//23K2001 - Muzammil
#include<iostream>
using namespace std;
class queue{
    private:
        int front,rear,size;
    public:
        string *arr;
        queue():front(-1),rear(-1),size(0),arr(nullptr){}
        queue(int s):front(-1),rear(-1),size(s){
            arr = new string[size];
            for(int i=0;i<size;i++)
                arr[i]="!";
        }
        void enqueue(string s){
            if(isFull()){
                cout<<"Queue is full!"<<endl;
                return;
            }
            else if(isEmpty())
                front = rear = 0;
            else
                rear++;
            arr[rear]=s;
            cout<<s<<" has been added to the queue."<<endl;
        }
        void dequeue(){
            if(isEmpty()){
                cout<<"Queue is empty!"<<endl;
                return;
            }
            else if(front == rear)
                front = rear = -1;
            else
                front++;
        }
        string atFront(){
            if(isEmpty()){
                cout<<"Queue is empty!"<<endl;
                return "!";
            }
            return arr[front];
```

```
        }
        bool isEmpty(){
            if(front==-1 || front>rear)
                return true;
            return false;
        }
        bool isFull(){
            if(rear==size-1)
                return true;
            return false;
        }
        void display(){
            if(isEmpty()){
                cout<<"Queue is empty!"<<endl;
                return;
            }
            cout<<endl<<"Patrons in waiting queue are: "<<endl;
            for(int i=front;i<=rear;i++){
                cout<<i<<". "<<arr[i]<<endl;
            }
        }
        ~queue(){ delete[] arr; }
};

const int maxPatrons = 15;

int main(){
    queue patrons(maxPatrons);
    int choice;
    string name;
    do{
        cout<<endl<<"\t\tFAST NU KHI LIBRARY"<<endl;
        cout<<"1. Add patron to queue"<<endl;
        cout<<"2. Remove patron from queue"<<endl;
        cout<<"3. Display patrons in queue"<<endl;
        cout<<"4. Exit"<<endl;
        cout<<"Enter your choice: ";
        cin>>choice;

        switch(choice){
            case 1:{
                cout<<"Enter patron's name: ";
                cin>>name;
                patrons.enqueue(name);
```

```
                break; }
            case 2:{
                if(!patrons.isEmpty()){
                    cout<<patrons.atFront()<<"'s transaction is
completed."<<endl;
                    patrons.dequeue();
                }
                else
                    cout<<"No patrons in the queue."<<endl;
                break; }
            case 3:{
                if(patrons.isEmpty())
                    cout<<"The queue is empty."<<endl;
                else
                    patrons.display();
                break; }
            case 4:
                break;
            default:
                cout<<"Invalid choice! Please try again."<<endl;
        }
    } while(choice != 4);

    return 0;
}
```

```
                FAST NU KHI LIBRARY
1. Add patron to queue
2. Remove patron from queue
3. Display patrons in queue
4. Exit
Enter your choice: 1
Enter patron's name: Muzammil
Muzammil has been added to the queue.

                FAST NU KHI LIBRARY
1. Add patron to queue
2. Remove patron from queue
3. Display patrons in queue
4. Exit
Enter your choice: 1
Enter patron's name: Talha
Talha has been added to the queue.

                FAST NU KHI LIBRARY
1. Add patron to queue
2. Remove patron from queue
3. Display patrons in queue
4. Exit
Enter your choice: 1
Enter patron's name: Nouman
Nouman has been added to the queue.
```

```
                FAST NU KHI LIBRARY
1. Add patron to queue
2. Remove patron from queue
3. Display patrons in queue
4. Exit
Enter your choice: 1
Enter patron's name: Farooq
Farooq has been added to the queue.

                FAST NU KHI LIBRARY
1. Add patron to queue
2. Remove patron from queue
3. Display patrons in queue
4. Exit
Enter your choice: 3

Patrons in waiting queue are:
0. Muzammil
1. Talha
2. Nouman
3. Farooq

                FAST NU KHI LIBRARY
1. Add patron to queue
2. Remove patron from queue
3. Display patrons in queue
4. Exit
Enter your choice: 2
Muzammil's transaction is completed.
```

```
                FAST NU KHI LIBRARY
1. Add patron to queue
2. Remove patron from queue
3. Display patrons in queue
4. Exit
Enter your choice: 2
Talha's transaction is completed.

                FAST NU KHI LIBRARY
1. Add patron to queue
2. Remove patron from queue
3. Display patrons in queue
4. Exit
Enter your choice: 3

Patrons in waiting queue are:
2. Nouman
3. Farooq

                FAST NU KHI LIBRARY
1. Add patron to queue
2. Remove patron from queue
3. Display patrons in queue
4. Exit
Enter your choice: 2
Nouman's transaction is completed.
```

```
                FAST NU KHI LIBRARY
1. Add patron to queue
2. Remove patron from queue
3. Display patrons in queue
4. Exit
Enter your choice: 2
Farooq's transaction is completed.

                FAST NU KHI LIBRARY
1. Add patron to queue
2. Remove patron from queue
3. Display patrons in queue
4. Exit
Enter your choice: 2
No patrons in the queue.

                FAST NU KHI LIBRARY
1. Add patron to queue
2. Remove patron from queue
3. Display patrons in queue
4. Exit
Enter your choice: 3
The queue is empty.

                FAST NU KHI LIBRARY
1. Add patron to queue
2. Remove patron from queue
3. Display patrons in queue
4. Exit
Enter your choice: 4
PS F:\Semester Material - Muzammil\FAS
```