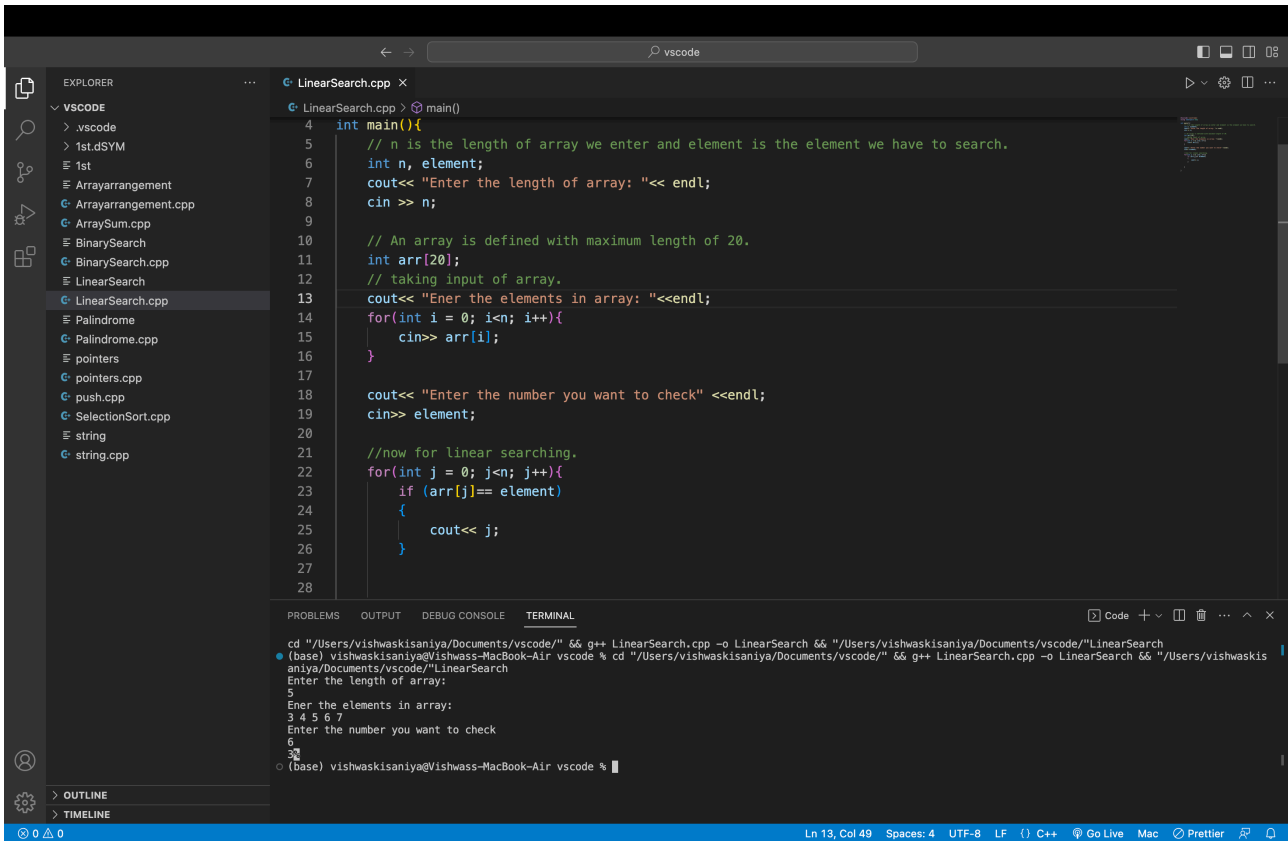


Name: Vishwas Kisaniya  
SID: 22106027  
Branch: CSE(DS)

Linear search:



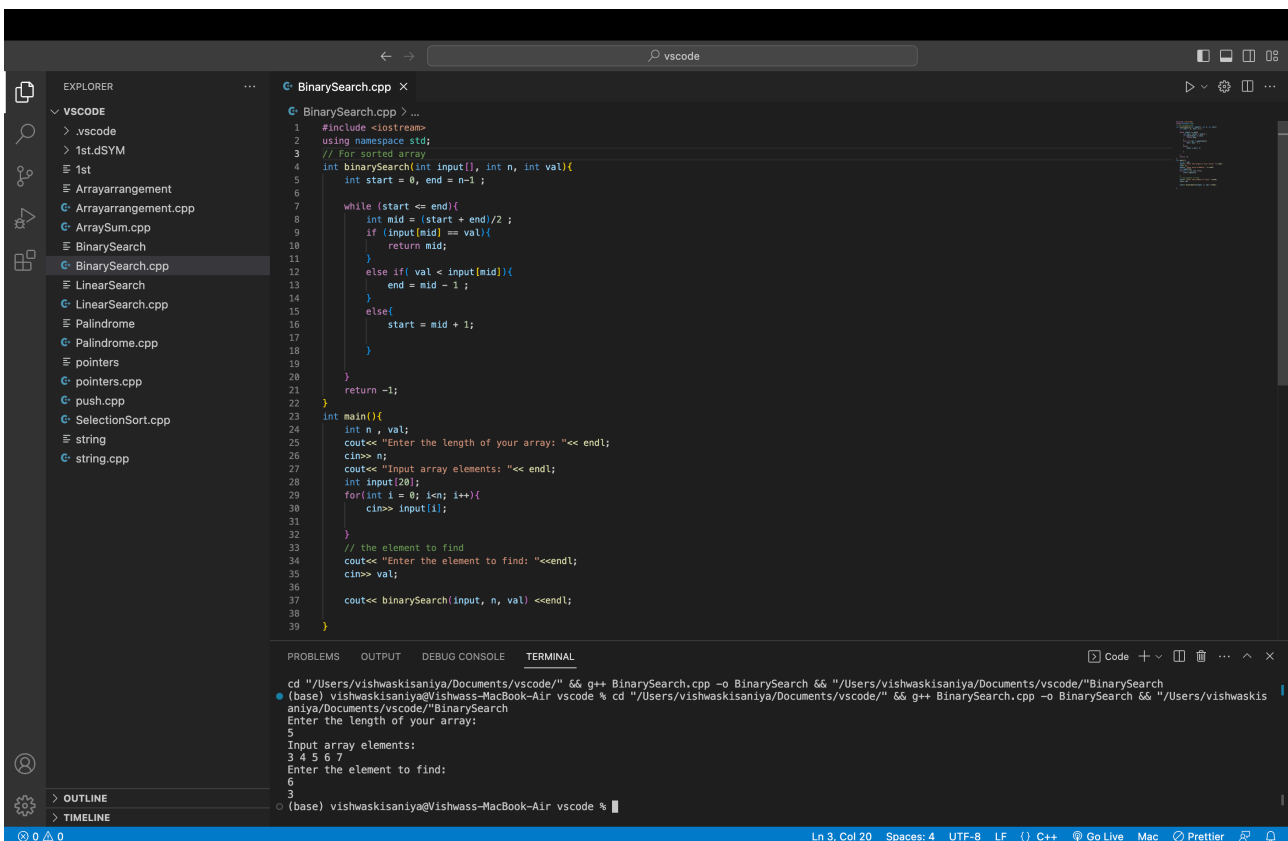
The screenshot shows the VS Code editor with the file `LinearSearch.cpp` open. The code implements a linear search algorithm. The terminal shows the execution of the program, where the user enters the length of the array (5), the elements (3 4 5 6 7), and the number to check (3). The program outputs the index of the element found.

```
4 int main(){
5     // n is the length of array we enter and element is the element we have to search.
6     int n, element;
7     cout<< "Enter the length of array: "<< endl;
8     cin >> n;
9
10    // An array is defined with maximum length of 20.
11    int arr[20];
12    // taking input of array.
13    cout<< "Enter the elements in array: "<< endl;
14    for(int i = 0; i<n; i++){
15        cin>> arr[i];
16    }
17
18    cout<< "Enter the number you want to check" << endl;
19    cin>> element;
20
21    //now for linear searching.
22    for(int j = 0; j<n; j++){
23        if (arr[j]== element)
24        {
25            cout<< j;
26        }
27    }
28 }
```

Terminal Output:

```
cd "/Users/vishwaskisaniya/Documents/vscode/" && g++ LinearSearch.cpp -o LinearSearch && "/Users/vishwaskisaniya/Documents/vscode/"LinearSearch
(base) vishwaskisaniya@Vishwas-MacBook-Air ~ % cd "/Users/vishwaskisaniya/Documents/vscode/" && g++ LinearSearch.cpp -o LinearSearch && "/Users/vishwaskisaniya/Documents/vscode/"LinearSearch
Enter the length of array:
5
Enter the elements in array:
3 4 5 6 7
Enter the number you want to check:
3
3
(base) vishwaskisaniya@Vishwas-MacBook-Air ~ %
```

Binary search:



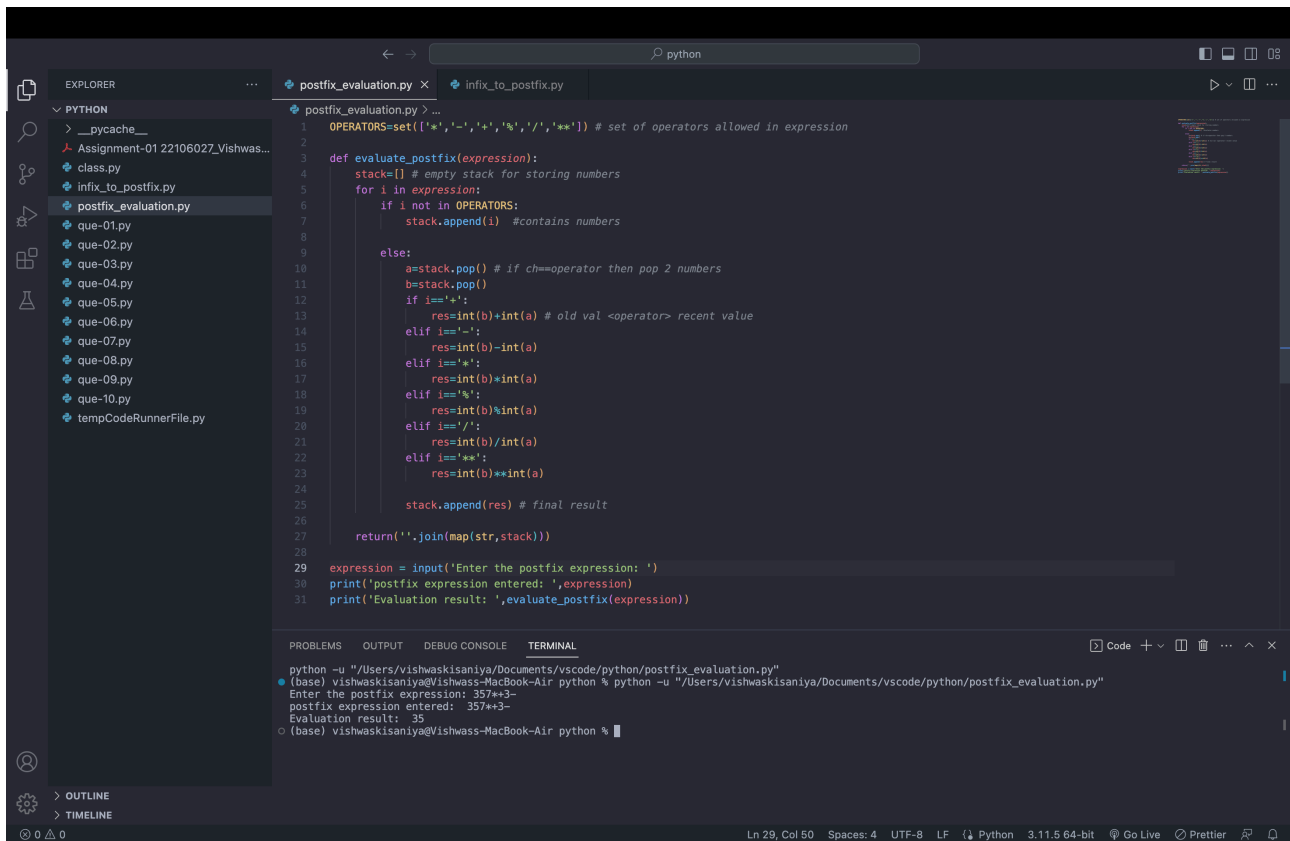
The screenshot shows the VS Code editor with the file `BinarySearch.cpp` open. The code implements a binary search algorithm. The terminal shows the execution of the program, where the user enters the length of the array (5), the elements (3 4 5 6 7), and the element to find (3). The program outputs the index of the element found.

```
1 #include <iostream>
2 using namespace std;
3 // For sorted array
4 int binarySearch(int input[], int n, int val){
5     int start = 0, end = n-1 ;
6
7     while (start <= end){
8         int mid = (start + end)/2 ;
9         if (input[mid] == val){
10             return mid;
11         }
12         else if (val < input[mid]){
13             end = mid - 1 ;
14         }
15         else{
16             start = mid + 1;
17         }
18     }
19     return -1;
20 }
21
22 int main(){
23     int n, val;
24     cout<< "Enter the length of your array: "<< endl;
25     cin>> n;
26     cout<< "Input array elements: "<< endl;
27     int input[20];
28     for(int i = 0; i<n; i++){
29         cin>> input[i];
30     }
31
32     // the element to find
33     cout<< "Enter the element to find: "<< endl;
34     cin>> val;
35
36     cout<< binarySearch(input, n, val) << endl;
37 }
38
39 }
```

Terminal Output:

```
cd "/Users/vishwaskisaniya/Documents/vscode/" && g++ BinarySearch.cpp -o BinarySearch && "/Users/vishwaskisaniya/Documents/vscode/"BinarySearch
(base) vishwaskisaniya@Vishwas-MacBook-Air ~ % cd "/Users/vishwaskisaniya/Documents/vscode/" && g++ BinarySearch.cpp -o BinarySearch && "/Users/vishwaskisaniya/Documents/vscode/"BinarySearch
Enter the length of your array:
5
Input array elements:
3 4 5 6 7
Enter the element to find:
3
3
(base) vishwaskisaniya@Vishwas-MacBook-Air ~ %
```

## Postfix evaluation:

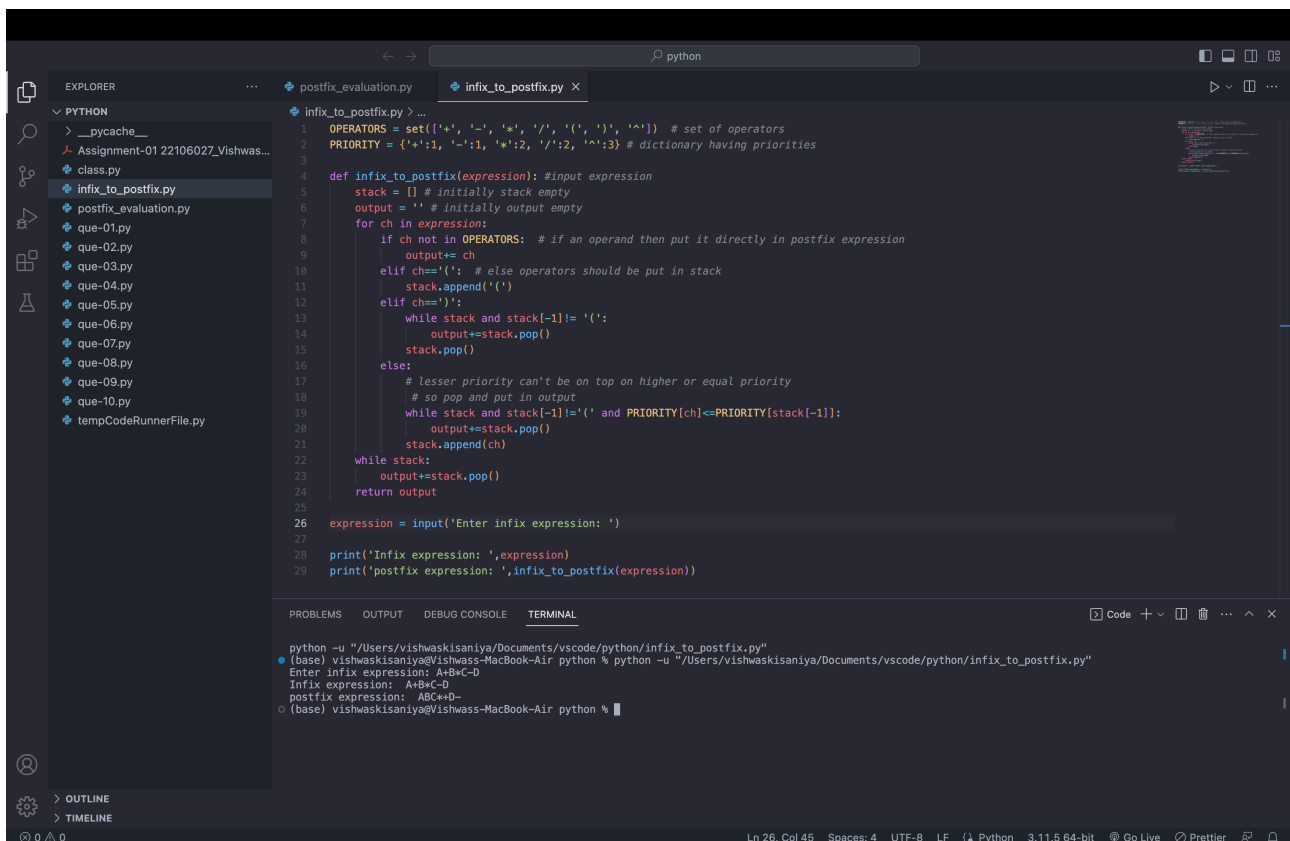


The screenshot shows a VS Code editor with a file explorer on the left containing a 'PYTHON' folder with various files. The main editor displays the code for 'postfix\_evaluation.py'. The code defines a set of operators and a function 'evaluate\_postfix' that processes a postfix expression using a stack. The terminal at the bottom shows the command 'python -u "/Users/vishwaskisaniya/Documents/vscode/python/postfix\_evaluation.py"' and the output: 'Enter the postfix expression: 357+\*3- Evaluation result: 35'.

```
1 OPERATORS=set(['+', '-', '*', '/', '^']) # set of operators allowed in expression
2
3
4 def evaluate_postfix(expression):
5     stack=[] # empty stack for storing numbers
6     for i in expression:
7         if i not in OPERATORS:
8             stack.append(i) #contains numbers
9         else:
10            a=stack.pop() # if ch==operator then pop 2 numbers
11            b=stack.pop()
12            if i=='+':
13                res=int(b)+int(a) # old val <operator> recent value
14            elif i=='-':
15                res=int(b)-int(a)
16            elif i=='*':
17                res=int(b)*int(a)
18            elif i=='/':
19                res=int(b)/int(a)
20            elif i=='^':
21                res=int(b)**int(a)
22            stack.append(res) # final result
23
24    return ''.join(map(str,stack))
25
26
27
28
29 expression = input('Enter the postfix expression: ')
30 print('postfix expression entered: ',expression)
31 print('Evaluation result: ',evaluate_postfix(expression))
```

python -u "/Users/vishwaskisaniya/Documents/vscode/python/postfix\_evaluation.py"  
(base) vishwaskisaniya@Vishwas-MacBook-Air python % python -u "/Users/vishwaskisaniya/Documents/vscode/python/postfix\_evaluation.py"  
Enter the postfix expression: 357+\*3-  
postfix expression entered: 357+\*3-  
Evaluation result: 35  
(base) vishwaskisaniya@Vishwas-MacBook-Air python %

## Infix to postfix conversion:



The screenshot shows a VS Code editor with a file explorer on the left. The main editor displays the code for 'infix\_to\_postfix.py'. The code defines a set of operators and their priorities, and a function 'infix\_to\_postfix' that converts an infix expression to postfix using a stack. The terminal at the bottom shows the command 'python -u "/Users/vishwaskisaniya/Documents/vscode/python/infix\_to\_postfix.py"' and the output: 'Enter infix expression: A+B\*C-D Infix expression: A+B\*C-D postfix expression: ABC+D-'.

```
1 OPERATORS = set(['+', '-', '*', '/', '^']) # set of operators
2 PRIORITY = {'+':1, '-':1, '*':2, '/':2, '^':3} # dictionary having priorities
3
4 def infix_to_postfix(expression): #input expression
5     stack = [] # initially stack empty
6     output = '' # initially output empty
7     for ch in expression:
8         if ch not in OPERATORS: # if an operand then put it directly in postfix expression
9             output+= ch
10        elif ch=='(': # else operators should be put in stack
11            stack.append('(')
12        elif ch==')':
13            while stack and stack[-1]!='(':
14                output+=stack.pop()
15            stack.pop()
16        else:
17            # lesser priority can't be on top on higher or equal priority
18            # so pop and put in output
19            while stack and stack[-1]!='(' and PRIORITY[ch]<=PRIORITY[stack[-1]]:
20                output+=stack.pop()
21            stack.append(ch)
22    while stack:
23        output+=stack.pop()
24    return output
25
26 expression = input('Enter infix expression: ')
27
28 print('Infix expression: ',expression)
29 print('postfix expression: ',infix_to_postfix(expression))
```

python -u "/Users/vishwaskisaniya/Documents/vscode/python/infix\_to\_postfix.py"  
(base) vishwaskisaniya@Vishwas-MacBook-Air python % python -u "/Users/vishwaskisaniya/Documents/vscode/python/infix\_to\_postfix.py"  
Enter infix expression: A+B\*C-D  
Infix expression: A+B\*C-D  
postfix expression: ABC+D-  
(base) vishwaskisaniya@Vishwas-MacBook-Air python %

Circular queue implémentation:

Code:

```
#include <iostream>

using namespace std;

class CircularQueue {
private:
    int *arr;
    int front;
    int rear;
    int size;
    int capacity;

public:
    CircularQueue(int capacity) {
        this->capacity = capacity;
        arr = new int[capacity];
        front = rear = -1;
        size = 0;
    }

    ~CircularQueue() {
        delete[] arr;
    }

    bool isEmpty() {
        return size == 0;
    }

    bool isFull() {
        return size == capacity;
    }

    void enqueue(int value) {
        if (isFull()) {
            cout << "Queue is full. Cannot enqueue." << endl;
            return;
        }

        if (isEmpty()) {
            front = rear = 0;
        } else {
            rear = (rear + 1) % capacity;
        }

        arr[rear] = value;
        size++;
    }

    void dequeue() {
        if (isEmpty()) {
            cout << "Queue is empty. Cannot dequeue." << endl;
            return;
        }

        if (front == rear) {
            front = rear = -1;
        }
    }
}
```

```

    } else {
        front = (front + 1) % capacity;
    }

    size--;
}

int peek() {
    if (isEmpty()) {
        cout << "Queue is empty. Cannot peek." << endl;
        return -1;
    }

    return arr[front];
}

void display() {
    if (isEmpty()) {
        cout << "Queue is empty." << endl;
        return;
    }

    int i = front;
    do {
        cout << arr[i] << " ";
        i = (i + 1) % capacity;
    } while (i != (rear + 1) % capacity);

    cout << endl;
}

};

int main() {
    CircularQueue queue(5);

    queue.enqueue(1);
    queue.enqueue(2);
    queue.enqueue(3);
    queue.enqueue(4);
    queue.enqueue(5);

    cout << "Queue elements: ";
    queue.display();

    cout << "Dequeue: ";
    queue.dequeue();
    queue.display();

    cout << "Peek: " << queue.peek() << endl;

    return 0;
}

```

Output:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
cd "/Users/vishwaskisaniya/Documents/vscode/" && g++ circular_queue.cpp -o circular_queue && "/Users/vishwaskisaniya/Documents/vscode/"circular_queue
(base) vishwaskisaniya@Vishwass-MacBook-Air vscode % cd "/Users/vishwaskisaniya/Documents/vscode/" && g++ circular_queue.cpp -o circular_queue && "/Users/vishwa
skisaniya/Documents/vscode/"circular_queue
Queue elements: 1 2 3 4 5
Dequeue: 2 3 4 5
Peek: 2
(base) vishwaskisaniya@Vishwass-MacBook-Air vscode %

```

Linear queue implementation:

Code:

```
#include <iostream>

using namespace std;

class LinearQueue {
private:
    int *arr;
    int front;
    int rear;
    int size;
    int capacity;

public:
    LinearQueue(int capacity) {
        this->capacity = capacity;
        arr = new int[capacity];
        front = rear = -1;
        size = 0;
    }

    ~LinearQueue() {
        delete[] arr;
    }

    bool isEmpty() {
        return size == 0;
    }

    bool isFull() {
        return size == capacity;
    }

    void enqueue(int value) {
        if (isFull()) {
            cout << "Queue is full. Cannot enqueue." << endl;
            return;
        }

        if (isEmpty()) {
            front = rear = 0;
        } else {
            rear++;
        }

        arr[rear] = value;
        size++;
    }

    void dequeue() {
        if (isEmpty()) {
            cout << "Queue is empty. Cannot dequeue." << endl;
            return;
        }

        if (front == rear) {
            front = rear = -1;
        }
    }
}
```

```

    } else {
        front++;
    }

    size--;
}

int peek() {
    if (isEmpty()) {
        cout << "Queue is empty. Cannot peek." << endl;
        return -1;
    }

    return arr[front];
}

void display() {
    if (isEmpty()) {
        cout << "Queue is empty." << endl;
        return;
    }

    for (int i = front; i <= rear; i++) {
        cout << arr[i] << " ";
    }

    cout << endl;
}

};

int main() {
    LinearQueue queue(5);

    queue.enqueue(1);
    queue.enqueue(2);
    queue.enqueue(3);
    queue.enqueue(4);
    queue.enqueue(5);

    cout << "Queue elements: ";
    queue.display();

    cout << "Dequeue: ";
    queue.dequeue();
    queue.display();

    cout << "Peek: " << queue.peek() << endl;

    return 0;
}

```

Output:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
cd "/Users/vishwaskisaniya/Documents/vscode/" && g++ linear_queue.cpp -o linear_queue && "/Users/vishwaskisaniya/Documents/vscode/"linear_queue
(base) vishwaskisaniya@Vishwass-MacBook-Air vscode % cd "/Users/vishwaskisaniya/Documents/vscode/" && g++ linear_queue.cpp -o linear_queue && "/Users/vishwaskisaniya/Documents/vscode/"linear_queue
Queue elements: 22 32 54 86 30
Dequeue: 32 54 86 30
Peek: 32
(base) vishwaskisaniya@Vishwass-MacBook-Air vscode %

```

