

Table of Content

<u>Prog. No.</u>	<u>Program Name</u>	<u>Page no.</u>	<u>Remarks</u>
1	Arithmetical Operations /Greatest & Least value /Odd Even /Sorting element	1 – 7	
2	Banking Transection	8 – 15	
3	Array Operations	16 - 23	
4	Linar Search	24 - 26	
5	Binary Search	27 – 29	
6	Stack Operations (without using switch case)	30 – 36	
7	Stack Operations	37 – 43	
8	Queue Operations	44 – 50	
9	Infix Expression to Postfix Expression	51 – 55	
10	Circular Queue	56 – 61	
00	Infix Expression to Prefix Expression	62 – 66	
11	Double Ended Queue	67 – 75	
12	Linked List	76 – 89	
13	Doubly Linked List	90 - 102	
14	Bubble Sort	103 - 105	
15	Insertion Sort	106 – 107	
16	Merge Sort	108 - 110	
17	Selection Sort	111 - 112	
18	Quick Sort	113 - 115	

Program No. 1**: (Aim) مقصد**

Write a C++ Program to Perform the following operations:

1. Arithmetical Operations
2. Greatest and Least value
3. Odd or Even Numbers
4. Sorting of elements using arrays (Ascending & Descending)
5. Exit

*Note: Include the above operations/tasks in a single program by using the functions and class with switch case.

: (Algorithm) الگورتھم

1. **Start** the program.
2. Create a **class Option** which contains functions for:
 - Arithmetic operations
 - Greatest & least value
 - Sorting (ascending and descending)
 - Odd/Even check
 - Displaying array elements
3. In **main()** function:
 - Initialize an object of Option class.
 - Display a **menu** with choices:
 - 1 → Arithmetic Operations
 - 2 → Greatest & Least
 - 3 → Sorting Ascending
 - 4 → Sorting Descending
 - 5 → Odd/Even
 - 6 → Exit
4. **Input user choice.**
5. Use **switch-case** to perform task based on choice:
 - If choice = 1 → Ask two numbers and perform addition, subtraction, multiplication, division.
 - If choice = 2 → Ask two numbers and find greatest and least.
 - If choice = 3 → Input array, sort elements in ascending order, display.
 - If choice = 4 → Input array, sort elements in descending order, display.
 - If choice = 5 → Input a number and check whether it is Odd or Even.
 - If choice = 6 → Exit program.
6. Repeat menu until user selects **Exit**.
7. **Stop** the program.

```

#include<iostream>
using namespace std;

class Option {
    int arr[200];
    int size;

public:
    Option() {
        size = 0;
    }

    void arithmetic() {
        int a,b;
        cout << "Enter two numbers: ";
        cin >> a >> b;
        cout << "Addition: " << a+b << endl;
        cout << "Subtraction: " << a-b << endl;
        cout << "Multiplication: " << a*b << endl;
        if(b != 0)
            cout << "Division: " << a/b << endl;
        else
            cout << "Undefine" << endl;
    }

    void greatest_least() {
        int a, b;
        cout << "Enter two numbers: ";
        cin >> a >> b;
        if(a > b)
            cout << "Greates: " << a << ", Least: " << b << endl;
        else
            cout << "Greates: " << b << ", Least: " << a << endl;
    }
}

```

```

void sort_ascending() {
    cout << "Enter size of array: ";
    cin >> size;
    cout << "Enter elements: \n";
    for(int i=0; i<size; i++) {
        cin >> arr[i];
    }
    for(int i=0; i<size; i++) {
        for(int j=i+1; j<size; j++) {
            if(arr[i] > arr[j]) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
    cout << "Elements in Ascending Order: \n";
    display();
}

```

```

void sort_descending() {
    cout << "Enter size of array: ";
    cin >> size;
    cout << "Enter elements: \n";
    for(int i=0; i<size; i++) {
        cin >> arr[i];
    }
    for(int i=0; i<size; i++) {
        for(int j=i+1; j<size; j++) {
            if(arr[i] < arr[j]) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}

```

```

        cout << "Elements in Descending Order: \n";
        display();
    }

    void odd_even() {
        int n;
        cout << "Enter a number: ";
        cin >> n;
        if(n % 2 == 0)
            cout << n << " is Even. \n";
        else
            cout << n << " is Odd. \n";
    }

    void display() {
        for(int i=0; i<size; i++) {
            cout << arr[i] << " ";
        }
        cout << endl;
    }
};

int main() {
    Option m;
    int choice = 0;
    while(choice != 6) {
        cout << "\nOptions---" << endl;
        cout << "1. Arithmetic Operations" << endl;
        cout << "2. Greatest & Least" << endl;
        cout << "3. Sorting in Ascending Order" << endl;
        cout << "4. Sorting in Descending Order" << endl;
        cout << "5. Odd/Even" << endl;
        cout << "6. Exit" << endl;

        cout << "\nEnter your choice: ";
        cin >> choice;
    }
}

```

```

switch(choice) {
    case 1: m.arithmetic(); break;
    case 2: m.greatest_least(); break;
    case 3: m.sort_ascending(); break;
    case 4: m.sort_descending(); break;
    case 5: m.odd_even(); break;
    case 6: cout << "Exiting..."; break;
    default : cout << "Invalid Input"; break;
}
}
return 0;
}

```

نتیجہ (Output):

```

C:\Users\Rashid Qamar\Docu... X + v
Options:
1. Arithmetic Operations
2. Greatest & Least
3. Sorting in Ascending Order
4. Sorting in Descending Order
5. Odd/Even
6. Exit

Enter your choice: 1
Enter two numbers: 20 10
Addition: 30
Subtraction: 10
Multiplication: 200
Division: 2

```

```

Options:
1. Arithmetic Operations
2. Greatest & Least
3. Sorting in Ascending Order
4. Sorting in Descending Order
5. Odd/Even
6. Exit

Enter your choice: 2
Enter two numbers: 20 10
Greatest: 20, Least: 10

```

Options:

1. Arithmetic Operations
2. Greatest & Least
3. Sorting in Ascending Order
4. Sorting in Descending Order
5. Odd/Even
6. Exit

Enter your choice: 3

Enter size of array: 5

Enter elements:

81 79 86 33 66

Elements in Ascending order:

33 66 79 81 86

Options:

1. Arithmetic Operations
2. Greatest & Least
3. Sorting in Ascending Order
4. Sorting in Descending Order
5. Odd/Even
6. Exit

Enter your choice: 4

Enter size of array: 4

Enter elements:

12 52 39 45

Elements in Descending order:

52 45 39 12

Options:

1. Arithmetic Operations
2. Greatest & Least
3. Sorting in Ascending Order
4. Sorting in Descending Order
5. Odd/Even
6. Exit

Enter your choice: 5

Enter a number: 38

38 is Even.

Options:

1. Arithmetic Operations
2. Greatest & Least
3. Sorting in Ascending Order
4. Sorting in Descending Order
5. Odd/Even
6. Exit

Enter your choice: 6

Exiting program...

Process exited after 232.5 seconds with return value 0
Press any key to continue . . . |

Program No. 2**المقصد (Aim) :**

Write a C++ Program to Perform the following Banking Transactions:

1. Login PIN
2. Balance Check
3. Cash/Amount Deposit
4. Cash/Amount Withdraw
5. Mini Statement
6. Change PIN
7. Exit

*Note: Include the above operations/tasks in a single program by using the functions and class with switch case.

الگورتھم (Algorithm) :

1. **Start** the program.
2. Create a class **bank** which contains functions for:
 - pinCreate() → to create login PIN
 - checkBalance() → to check current balance
 - deposit() → to deposit money into account
 - withdraw() → to withdraw money if balance is sufficient
 - miniStatement() → to display transaction history
 - changePin() → to update/change PIN
3. In **main()** function:
 - Initialize an object of **bank** class.
 - Display a menu with choices:
 - 1 → Create PIN
 - 2 → Check Balance
 - 3 → Deposit Amount
 - 4 → Withdraw Amount
 - 5 → Mini Statement
 - 6 → Change PIN
 - 7 → Exit
4. **Take input from user (choice).**
5. Use **switch-case** to perform task based on choice:
 - If choice = 1 → Call pinCreate() to create PIN.
 - If choice = 2 → Call checkBalance() after verifying PIN.
 - If choice = 3 → Call deposit() to add money and record transaction.

- If choice = 4 → Call withdraw() to withdraw money if sufficient balance.
- If choice = 5 → Call miniStatement() to display transaction history.
- If choice = 6 → Call changePin() to update PIN.
- If choice = 7 → Exit program.

6. Repeat menu until user selects **Exit**.

7. **Stop** the program.

کوڈ (Code):

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

class bank {
    float balance;
    int loginPin;

    string typeHistory[100];
    float amount[100];
    int count;

public :
    bank() {
        balance = 0;
        count = 0;
    }

    void pinCreate() {
        cout << "Create your pin: ";
        cin >> loginPin;
        cout << "Your PIN has been created Succesfully !!\n";
    }

    void checkBalance() {
        int pin;
        cout << "Enter your PIN: ";
        cin >> pin;
        if(pin == loginPin) {
```

```

        cout << "Your current balance is " << balance << endl;
    }
    else {
        cout << "You have Entered wrong PIN\n";
    }
}

```

```

void deposit() {
    int deposit, pin;
    cout << "Enter amount to deposit: ";
    cin >> deposit;
    cout << "Enter your PIN: ";
    cin >> pin;
    if(pin == loginPin) {
        balance += deposit;
        cout << "Your amount deposit Succesfully !!\n";

        typeHistory[count] = "Deposited";
        amount[count] = deposit;
        count++;
    }
    else {
        cout << "You have Entered wrong PIN\n";
    }
}

```

```

void withdraw() {
    int withdraw, pin;
    cout << "Enter amount to withdraw: ";
    cin >> withdraw;
    cout << "Enter your PIN: ";
    cin >> pin;
    if(pin == loginPin) {
        if(withdraw <= balance) {
            balance -= withdraw;
            cout << "Your amount Withdraw Successfully !!";

```

```

        typeHistory[count] = "Withdraw";
        amount[count] = withdraw;
        count++;
    }
    else {
        cout << "You have insufficient amount \n";
    }
}
else {
    cout << "You have Entered wrong PIN\n";
}
}

void miniStatement() {
    cout << "Mini Statement" << endl;
    cout << "-----" << endl;
    if(count ==0) {
        cout << "No Transection Found";
    }
    else {
        for(int i=0; i<count; i++) {
            cout <<i+1<<". "<<typeHistory[i]<<" Rs."<<amount[i]<<endl;
        }
    }
}

void changePin() {
    int pin, newPin;
    cout << "Enter your old PIN: ";
    cin >> pin;
    if(pin == loginPin) {
        cout << "Enter new PIN: ";
        cin >> newPin;
        loginPin = newPin;
        cout << "Your PIN has been changed Successfully !!\n";
    }
}

```

```

    }
    else {
        cout << "You have entered wrong PIN\n";
    }
}
};

int main() {
    bank b;
    int choice=0;
    while(choice != 7) {
        cout << "\nOptions: " << endl;
        cout << "-----" << endl;
        cout << "1. Login pin" << endl;
        cout << "2. Check Balance" << endl;
        cout << "3. Amount Deposit" << endl;
        cout << "4. Amount Withdraw" << endl;
        cout << "5. Mini Statement" << endl;
        cout << "6. Change Pin" << endl;
        cout << "7. Exit" << endl;

        cout << "\nEnter your choice: ";
        cin >> choice;

        switch(choice) {
            case 1: b.pinCreate(); break;
            case 2: b.checkBalance(); break;
            case 3: b.deposit(); break;
            case 4: b.withdraw(); break;
            case 5: b.miniStatement(); break;
            case 6: b.changePin(); break;
            case 7: cout << "Exiting..."; break;

        }
    }
    return 0;
}

```

نتیجہ (Output):

C:\Users\Rashid Qamar\Docu

+

v

Options:

1. Login pin
2. Check Balance
3. Amount Deposite
4. Amount Withdraw
5. Mini Statement
6. Change Pin
7. Exit

Enter your choice: 1

Create your pin: 3366

Your PIN has been created Succesfully !!

Options:

1. Login pin
2. Check Balance
3. Amount Deposite
4. Amount Withdraw
5. Mini Statement
6. Change Pin
7. Exit

Enter your choice: 2

Enter your PIN: 3366

Your current balance is 0

Options:

1. Login pin
2. Check Balance
3. Amount Deposite
4. Amount Withdraw
5. Mini Statement
6. Change Pin
7. Exit

Enter your choice: 3

Enter amount to deposite: 2000

Enter your PIN: 3366

Your amount deposite Succesfully !!

Options:

1. Login pin
2. Check Balance
3. Amount Deposit
4. Amount Withdraw
5. Mini Statement
6. Change Pin
7. Exit

Enter your choice: 4

Enter amount to withdraw: 1000

Enter your PIN: 3366

Your amount Withdraw Successfully !!

Options:

1. Login pin
2. Check Balance
3. Amount Deposit
4. Amount Withdraw
5. Mini Statement
6. Change Pin
7. Exit

Enter your choice: 5

Mini Statement

1. Deposited Rs.2000
2. Withdraw Rs.1000

Options:

1. Login pin
2. Check Balance
3. Amount Deposit
4. Amount Withdraw
5. Mini Statement
6. Change Pin
7. Exit

Enter your choice: 6

Enter your old PIN: 3366

Enter new PIN: 3456

Your PIN has been changed Successfully !!

Options:

-
1. Login pin
 2. Check Balance
 3. Amount Deposite
 4. Amount Withdraw
 5. Mini Statement
 6. Change Pin
 7. Exit

Enter your choice: 7

Exiting...

Process exited after 209.2 seconds with return value 0

Press any key to continue . . . |

Program No. 3**: (Aim) مقصد**

Write a C++ Program to perform the following operations on Arrays:

1. Insertion of an Element at a particular location/position
2. Deletion of a particular element
3. Searching for an element and displaying its location/index
4. Updating an Element in an existing array
5. Display the updated array
6. Exit

*Note: Include the above operations/tasks in a single program by using the functions and classes with a switch case.

: (Algorithm) الگورتھم

1. **Start** the program.
2. Create a class **Array** which contains functions for:
 - input() → to input array elements
 - insert() → to insert a new element at a given position
 - deleteEle() → to delete an element from array
 - search() → to search an element in array
 - update() → to update an element with new value
 - display() → to display array elements
3. In **main()** function:
 - Create an object of class **Array**.
 - Call input() to take initial array size and elements.
 - Display a **menu** with choices:
 - 1 → Insert an element
 - 2 → Delete an element
 - 3 → Search an element
 - 4 → Update array
 - 5 → Display array
 - 6 → Exit
4. **Take Input from user (choice).**
5. Use **switch-case** to perform task based on choice:
 - If choice = 1 → Call insert() and display updated array.
 - If choice = 2 → Call deleteEle() and display updated array.
 - If choice = 3 → Call search() to check if element exists and its position.
 - If choice = 4 → Call update() to replace element with new value.

- If choice = 5 → Call display() to show array elements.
- If choice = 6 → Exit program.

6. Repeat menu until user selects **Exit**.

7. **Stop** the program.

کوڈ (Code):

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

class Array {
    int a[200];
    int n;

public:
    Array() {
        n = 0;
    }

    void input() {
        cout << "Enter Array size: ";
        cin >> n;
        cout << "Enter numbers: \n";
        for (int i = 0; i < n; i++) {
            cin >> a[i];
        }
    }

    void insert() {
        int ele, pos;
        cout << "Enter element to insert: ";
        cin >> ele ;
        cout << "Enter position: ";
        cin >> pos;
        pos = pos-1;
        n++;
        for(int i=n-1; i>pos; i--) {
            a[i] = a[i-1];
```

```

    }
    a[pos] = ele;
    cout << "Updated Array: \n";
    display();
}

void deleteEle() {
    int ele, found=0;
    cout << "Enter element to delete: ";
    cin >> ele;
    for(int i=0; i<n; i++) {
        if(a[i] == ele) {
            found = 1;
            for(int j=i; j<n-1; j++) {
                a[j] = a[j+1];
            }
            n--;
            break;
        }
    }
    if(found == 1) {
        cout << "Updated Array: \n";
        display();
    } else {
        cout << "Element NOT found";
    }
}

void search() {
    int ele, flag=0, pos;
    cout << "Enter element to search: ";
    cin >> ele;
    for(int i=0; i<n; i++) {
        if(a[i] == ele) {
            flag = 1;
            pos = i;

```

```

        break;
    }
}
if(flag == 1) {
    cout << "Element found at "<<pos+1<<" position\n";
} else {
    cout << "Element NOT found\n";
}
}

void update() {
    int ele, update;
    cout << "Enter element to update: ";
    cin >> ele;
    cout << "Enter new element: ";
    cin >> update;
    for(int i=0; i<n; i++) {
        if(a[i] == ele) {
            a[i] = update;
        }
    }
    cout << "Updated Array: \n";
    display();
}

void display() {
    for(int i=0; i<n; i++) {
        cout << a[i] << " ";
    }
    cout << endl;
}

};

int main() {
    Array m;
    m.input() ;

```

```
int choice=0;
while(choice != 6) {
    cout << "\nOptions :< endl;
    cout << "-----" << endl;
    cout << "1. Insertion of an element" << endl;
    cout << "2. Deletion of an element" << endl;
    cout << "3. Searching of an element" << endl;
    cout << "4. Update Array" << endl;
    cout << "5. Display the updated array" << endl;
    cout << "6. Exit" << endl;

    cout << "\nEnter your choice: ";
    cin >> choice;

    switch(choice) {
        case 1: m.insert(); break;
        case 2: m.deleteEle(); break;
        case 3: m.search(); break;
        case 4: m.update(); break;
        case 5: m.display(); break;
        case 6: cout << "Exiting..."; break;
    }
}
return 0;
}
```

نتیجہ (Output):

C:\Users\Rashid Qamar\Docu ×

+

v

Enter Array size: 5

Enter numbers:

81 79 86 33 66

Options :

1. Insertion of an element
2. Deletion of an element
3. Searching of an element
4. Update Array
5. Display the updated array
6. Exit

Enter your choice: 1

Enter element to insert: 29

Enter position: 4

Updated Array:

81 79 86 29 33 66

Options :

1. Insertion of an element
2. Deletion of an element
3. Searching of an element
4. Update Array
5. Display the updated array
6. Exit

Enter your choice: 2

Enter element to delete: 29

Updated Array:

81 79 86 33 66

Options :

1. Insertion of an element
2. Deletion of an element
3. Searching of an element
4. Update Array
5. Display the updated array
6. Exit

Enter your choice: 3

Enter element to search: 86

Element found at 3 position

Options :

1. Insertion of an element
2. Deletion of an element
3. Searching of an element
4. Update Array
5. Display the updated array
6. Exit

Enter your choice: 4

Enter element to update: 86

Enter new element: 89

Updated Array:

81 79 89 33 66

Options :

1. Insertion of an element
2. Deletion of an element
3. Searching of an element
4. Update Array
5. Display the updated array
6. Exit

Enter your choice: 5

81 79 89 33 66

Options :

1. Insertion of an element
2. Deletion of an element
3. Searching of an element
4. Update Array
5. Display the updated array
6. Exit

Enter your choice: 6

Exiting...

Process exited after 253.1 seconds with return value 0

Press any key to continue . . . |

Program No. 4**مقصد (Aim) :**

Write a C++ Program to perform the Linear Search:

1. Accept the size of the array and the elements from the user.
2. Display the entered array.
3. Ask the user to enter an element to search.
4. If the element is found, display its index; otherwise, display a message that the element is not present.

الگورتھم (Algorithm) :

1. **Start** the program.
2. Input the number of elements **n**.
3. Declare an array **a[n]**.
4. Input **n** numbers into the array.
5. Display the entered array elements.
6. Input the element to be searched (**search**).
7. Initialize **found = 0**.
8. Repeat for **i = 0** to **n-1**:
 - If **a[i] == search**, then:
 - Set **found = 1**.
 - Store position = **i + 1**. //Bcz pos starting from 0.
 - **break**.
9. After loop:
 - If **found == 1** → Display "Element found at position ...".
 - Else → Display "Element NOT found".
10. **Stop** the program.

کوڈ (Code) :

```
#include<iostream>
using namespace std;
int main() {
    int n, search, found=0, position;
    cout << "Enter no. of element: ";
    cin >> n;
    int a[n];
    cout << "Enter "<< n << " numbers: "<< endl;
```

```
for(int i=0; i<n; i++) {  
    cin >> a[i];  
}  
cout << "\nEntered numbers: " << endl;  
for(int i=0; i<n; i++) {  
    cout << a[i] << " ";  
}  
cout << "\n\nEnter search element: ";  
cin >> search;  
  
for(int i=0; i<n; i++) {  
    if(search == a[i]) {  
        found = 1;  
        position = i+1;  
        break;  
    }  
}  
if(found == 1) {  
    cout << "Element found at " << position << " position";  
}  
else {  
    cout << "Element NOT found";  
}  
return 0;  
}
```

نتیجہ (Output):

```
C:\Users\Rashid Qamar\Docu  ×  +  v
Enter no. of element: 5
Enter 5 numbers:
81
79
86
33
66

Entered numbers:
81 79 86 33 66

Enter search element: 33
Element found at 4 position
-----
Process exited after 17.52 seconds with return value 0
Press any key to continue . . . |
```

Program No. 5**مقصد (Aim) :**

Write a C++ Program to perform the Binary Search:

1. Accept the size of the array and the elements from the user.
2. Display the entered array.
3. Ask the user to enter an element to search.
4. If the element is found, display its index; otherwise, display a message that the element is not present.

الگورتھم (Algorithm) :

1. **Start** the program.
2. Input the number of elements **n**.
3. Declare an array **a[n]**.
4. Input **n** numbers in **ascending order** into the array.
5. Display the entered numbers.
6. Input the element to be searched (search).
7. Initialize **start = 0**, **end = n-1**, **found = 0**.
8. Repeat while **start <= end**:
 - Calculate **mid = (start + end) / 2**.
 - If **search > a[mid]**, then set **start = mid + 1**.
 - Else if **search < a[mid]**, then set **end = mid - 1**.
 - Else (when **search == a[mid]**):
 - Set **found = 1**.
 - Break the loop.
9. After loop ends:
 - If **found == 1** → Display "Element found at position ...".
 - Else → Display "Element NOT found".
10. **Stop** the program.

کوڈ (Code) :

```
#include<iostream>
using namespace std;
int main() {
    int n, i, search, mid, found=0;
    cout << "Enter no. of element: ";
    cin >> n;
```

```

int a[n];
int start=0, end=n-1;
cout << "Enter number in Ascending Order: " << endl;
for (i=0; i<n; i++) {
    cin >> a[i];
}
cout << "\nEnter number: " << endl;
for (i=0; i<n; i++) {
    cout << a[i] << " ";
}
cout << "\nEnter search element: ";
cin >> search;

while(start <= end) {
    mid = (start+end)/2;
    if (search > a[mid]) {
        start = mid + 1;
    }
    else if(search < a[mid]) {
        end = mid - 1;
    }
    else {
        found = 1;
        break;
    }
}
if(found == 1) {
    cout << "\nElement found at " << mid+1 << " Position\n";
}
else {
    cout << "Element NOT found\n";
}
return 0;
}

```

نتیجہ (Output):

C:\Users\Rashid Qamar\Docu

+

v

```
Enter no. of element: 5
Enter number in Ascending Order:
81
79
86
33
66
```

```
Entered number:
81 79 86 33 66
```

```
Enter search element: 86
```

```
Element found at 3 Position
```

```
-----
Process exited after 17.34 seconds with return value 0
Press any key to continue . . . |
```

Program No. 6**المقصد (Aim) :**

Write a C++ program to implement basic stack operations (push, pop, peek, isEmpty, isFull, and display) without using a switch case.

الگورتهم (Algorithm) :

1. **Start** the program.
2. Define a constant MAX for maximum size of stack.
3. Declare an array stack[MAX] and initialize top = -1.
4. Display the menu of options:
 - 1 → Push
 - 2 → Pop
 - 3 → Peek
 - 4 → isEmpty?
 - 5 → isFull?
 - 6 → Display
 - 7 → Exit
5. **Take Input from user (choice).**
6. Perform operations according to choice:
 - **Case 1: Push**
 - If top == MAX-1 → Stack Overflow (cannot push).
 - Else increment top and insert element at stack[top].
 - **Case 2: Pop**
 - If top == -1 → Stack Underflow (cannot pop).
 - Else display stack[top] and decrement top.
 - **Case 3: Peek**
 - If top == -1 → Stack is Empty.
 - Else display stack[top].
 - **Case 4: isEmpty**
 - If top == -1 → Display "Stack is Empty".
 - Else "Stack is Not Empty".
 - **Case 5: isFull**
 - If top == MAX-1 → Display "Stack is Full".
 - Else "Stack is Not Full".
 - **Case 6: Display**
 - If top == -1 → Display "Stack is Empty".
 - Else print all elements from stack[top] down to stack[0].

- **Case 7: Exit**

- Stop the program.

7. Repeat steps until choice = 7.

8. **End** the program.

کوڈ (Code):

```
#include<iostream>
using namespace std;
#define MAX 10
int main() {
    int stack[MAX], top=-1;
    int choice, value;
    while(choice != 7) {
        cout << "\nOptions: " << endl;
        cout << "1. Push" << endl;
        cout << "2. Pop" << endl;
        cout << "3. Peek" << endl;
        cout << "4. isEmpty?" << endl;
        cout << "5. isFull?" << endl;
        cout << "6. Display" << endl;
        cout << "7. Exit" << endl;

        cout << "\nSelect any option: ";
        cin >> choice;

        if(choice == 1) {
            if(top == MAX-1) {
                cout << "Stack is Full, Can't Push\n";
            }
            else {
                cout << "Enter value to push: ";
                cin >> value;
                top++;
                stack[top] = value;
                cout << value << " pushed, Successfully! \n";
            }
        }
    }
}
```



```
}

else if(choice == 2) {
    if(top == -1) {
        cout << "stack is empty, Can't pop\n";
    }
    else {
        cout << stack[top] << " Popped, Successfully! \n";
        top--;
    }
}

else if(choice == 3) {
    if(top == -1) {
        cout << "Stack is empty \n";
    }
    else {
        cout << "Top element is " << stack[top];
    }
}

else if(choice == 4) {
    if(top == -1) {
        cout << "Stack is Empty \n";
    }
    else {
        cout << "Stack is not Empty \n";
    }
}

else if(choice == 5) {
    if(top == MAX-1) {
        cout << "Stack is Full \n";
    }
    else {
        cout << "Stack is Not Full \n";
    }
}
```

```
        }  
    }  
  
    else if(choice == 6) {  
        if(top == -1) {  
            cout << "Stack is Empty \n";  
        }  
        else {  
            cout << "Stack element are: \n";  
            for(int i=top; i>=0; i--) {  
                cout << stack[i] << " ";  
            }  
            cout << endl;  
        }  
    }  
  
    else if(choice == 7) {  
        cout << "Exiting...\n";  
    }  
}  
return 0;  
}
```

نتیجہ (Output):

C:\Users\Rashid Qamar\Docu

+

v

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 1

Enter value to push: 10

10 pushed, Successfully!

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 1

Enter value to push: 20

20 pushed, Successfully!

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 2

20 Popped, Successfully!

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 3

Top element is 10

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 4

Stack is not Empty

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 5

Stack is Not Full

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 6

Stack element are:

10

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 7

Exiting...

Process exited after 320.1 seconds with return value 0

Press any key to continue . . . |

Program No. 7**المقصد (Aim) :**

Write a C++ program to implement basic stack operations using a switch case.

1. Push
2. Pop
3. Peek
4. isEmpty
5. isFull
6. Display
7. Exit

الغورتهم (Algorithm) :

1. **Start** the program.
2. Define a constant MAX for maximum size of stack.
3. Declare an array stack[MAX] and initialize top = -1.
4. Display the menu of options:
 - 1 → Push
 - 2 → Pop
 - 3 → Peek
 - 4 → isEmpty?
 - 5 → isFull?
 - 6 → Display
 - 7 → Exit
5. Take Input from user (choice).
6. Perform operations according to choice:
 - **Case 1: Push**
 - If top == MAX-1 → Stack Overflow (cannot push).
 - Else increment top and insert element at stack[top].
 - **Case 2: Pop**
 - If top == -1 → Stack Underflow (cannot pop).
 - Else display stack[top] and decrement top.
 - **Case 3: Peek**
 - If top == -1 → Stack is Empty.
 - Else display stack[top].
 - **Case 4: isEmpty**
 - If top == -1 → Display "Stack is Empty".
 - Else "Stack is Not Empty".

- **Case 5: isFull**
 - If `top == MAX-1` → Display "Stack is Full".
 - Else "Stack is Not Full".
 - **Case 6: Display**
 - If `top == -1` → Display "Stack is Empty".
 - Else print all elements from `stack[top]` down to `stack[0]`.
 - **Case 7: Exit**
 - Stop the program.
7. Repeat steps until choice = 7.
8. **End** the program.

کوڈ (Code):

```
#include<iostream>
using namespace std;
#define MAX 10
int main() {
    int stack[MAX], top=-1;
    int choice=0, value;
    while(choice != 7) {
        cout << "\nOptions: " << endl;
        cout << "1. Push" << endl;
        cout << "2. Pop" << endl;
        cout << "3. Peek" << endl;
        cout << "4. isEmpty?" << endl;
        cout << "5. isFull?" << endl;
        cout << "6. Display" << endl;
        cout << "7. Exit" << endl;

        cout << "\nSelect any option: ";
        cin >> choice;
        switch(choice) {
            case 1:{
                if(top == MAX-1) {
                    cout << "Stack is Full, Can't Push\n";
                }
                else {
```

```

        cout << "Enter value to push: ";
        cin >> value;
        top++;
        stack[top] = value;
        cout << value << " pushed, Successfully! \n";
    }
    break;
}

case 2:{
    if(top == -1) {
        cout << "stack is empty, Can't pop\n";
    }
    else {
        cout << stack[top] << " Popped, Successfully! \n";
        top--;
    }
    break;
}

case 3:{
    if(top == -1) {
        cout << "Stack is empty \n";
    }
    else {
        cout << "Top element is " << stack[top];
    }
    break;
}

case 4:{
    if(top == -1) {
        cout << "Stack is Empty \n";
    }
    else {
        cout << "Stack is not Empty \n";
    }
}

```



```

        }
        break;
    }

    case 5:{
        if(top == MAX-1) {
            cout << "Stack is Full \n";
        }
        else {
            cout << "Stack is Not Full \n";
        }
        break;
    }

    case 6:{
        if(top == -1) {
            cout << "Stack is Empty \n";
        }
        else {
            cout << "Stack element are: \n";
            for(int i=top; i>=0; i--) {
                cout << stack[i] << " ";
            }
            cout << endl;
        }
        break;
    }

    case 7:{
        cout << "Exiting...\n";
        break;
    }

}

return 0;
}

```

نتیجہ (Output):

C:\Users\Rashid Qamar\Docu ×

+ v

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 1

Enter value to push: 10

10 pushed, Successfully!

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 1

Enter value to push: 20

20 pushed, Successfully!

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 2

20 Popped, Successfully!

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 3
Top element is 10

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 4
Stack is not Empty

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 5
Stack is Not Full

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 6
Stack element are:
10

Options:

1. Push
2. Pop
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 7

Exiting...

Process exited after 172.2 seconds with return value 0
Press any key to continue . . . |

Program No. 8**المقصد (Aim) :**

Write a C++ Program to Perform basic Queue operations using a switch case.

1. Insertion
2. Deletion
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

الگورتهم (Algorithm) :

1. **Start** the program.
2. Define a class **Queue** with:
 - An array Q[] of fixed size.
 - Variables: front, rear, and n (size of array).
 - Functions for:
 - insert() → to add element.
 - delete() → to remove element.
 - peek() → to show front element.
 - empty() → to check if queue is empty.
 - full() → to check if queue is full.
 - display() → to show all queue elements.
3. **Initialize Queue:**
 - Set front = -1, rear = -1, n = 5.
4. **Insert Operation:**
 - If rear == n-1, print "Queue is Full".
 - Else, if front == -1, set front = 0.
 - Increment rear and place new element at Q[rear].
5. **Delete Operation:**
 - If rear == -1 OR front > rear, print "Queue is Empty".
 - Else, print Q[front] and increment front.
6. **Peek Operation:**
 - If queue is empty, display message.
 - Else, show Q[front].
7. **isEmpty Check:**
 - If front == -1 OR front > rear, then queue is empty.
 - Else, queue is not empty.

8. isFull Check:

- If $\text{rear} == n-1$, then queue is full.
- Else, not full.

9. Display Operation:

- If queue is empty, show message.
- Else, print all elements from front to rear.

10. In main() function:

- Create an object of Queue class.
- Display menu with options:
 - 1 → Insertion
 - 2 → Deletion
 - 3 → Peek
 - 4 → isEmpty
 - 5 → isFull
 - 6 → Display
 - 7 → Exit
- Input user's choice.
- Use switch-case to perform corresponding operation.

11. **Repeat menu** until user chooses Exit.

12. **Stop** the program.

کوڈ (Code):

```
#include<iostream>
using namespace std;
class Queue {
    int Q[5];
    int n, front, rear;
public:
    Queue() {
        n=5, front=-1, rear=-1;
    }

    void insert() {
        int value;
        if(rear == n-1) {
            cout << "Queue is Full \n";
        }
    }
}
```

```

        else {
            if(front == -1) {
                front = 0;
            }
            cout << "Enter element to insert: ";
            cin >> value;
            rear++;
            Q[rear] = value;
            cout << value << " Inserted, Successfully \n";
        }
    }

void Delete() {
    if(rear == -1 || front > rear) {
        cout << "Queue is empty \n";
    }
    else {
        cout << "Element deleted from Queue is :" << Q[front] << endl;
        front++;
    }
}

void peek() {
    if (front == -1 || front > rear) {
        cout << "Queue is Empty, nothing to peek\n";
    }
    else {
        cout << "Front element is: " << Q[front] << endl;
    }
}

void empty() {
    if (front == -1 || front > rear)
        cout << "Queue is empty \n";
    else
        cout << "Queue is NOT empty \n";
}

```

```

    }

    void full() {
        if (rear == n - 1)
            cout << "Queue is full \n";
        else
            cout << "Queue is NOT full \n";
    }

    void display() {
        if(front == -1 || front > rear) {
            cout << "Queue is Empty \n";
        }
        else {
            cout << "Queue elements are: \n";
            for(int i=front; i<=rear; i++) {
                cout << Q[i] << " ";
            }
            cout << endl;
        }
    }
}

};

int main() {
    Queue m;
    int choice=0;
    while(choice != 7) {
        cout << "\nOptions: " << endl;
        cout << "1. Insertion" << endl;
        cout << "2. Deletion" << endl;
        cout << "3. Peek" << endl;
        cout << "4. isEmpty?" << endl;
        cout << "5. isFull?" << endl;
        cout << "6. Display" << endl;
        cout << "7. Exit" << endl;
    }
}

```



```

        cout << "\nSelect any option: ";
        cin >> choice;

        switch(choice) {
            case 1: m.insert(); break;
            case 2: m.Delete(); break;
            case 3: m.peek(); break;
            case 4: m.empty(); break;
            case 5: m.full(); break;
            case 6: m.display(); break;
            case 7: cout << "Exiting..."; break;
        }
    }
    return 0;
}

```

نتیجہ (Output):

```

Options:
1. Insertion
2. Deletion
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 1
Enter element to insert: 10
10 Inserted, Successfully

Options:
1. Insertion
2. Deletion
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 1
Enter element to insert: 20
20 Inserted, Successfully

```

Options:

1. Insertion
2. Deletion
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 2

Element deleted from Queue is :10

Options:

1. Insertion
2. Deletion
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 3

Front element is: 20

Options:

1. Insertion
2. Deletion
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 4

Queue is NOT empty

Options:

1. Insertion
2. Deletion
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 5

Queue is NOT full

Options:

1. Insertion
2. Deletion
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 6

Queue elements are:

20

Options:

1. Insertion
2. Deletion
3. Peek
4. isEmpty?
5. isFull?
6. Display
7. Exit

Select any option: 7

Exiting...

Process exited after 202.1 seconds with return value 0

Press any key to continue . . . |

Program No. 9**المقصد (Aim) :**

Write a C++ Program to convert the infix expression to a postfix expression using a Stack.

الگورتھم (Algorithm) :

1. **Start** the program.
2. **Initialize a stack** to hold operators.
 - top = -1 initially (empty stack).
3. **Take Input the infix expression** from the user.
4. For each symbol of the infix expression (from left to right):
 - If **operand** → add it directly to postfix expression.
 - If **'(' (left parenthesis)** → push it onto the stack.
 - If **')' (right parenthesis)** →
 - Repeatedly pop operators from stack and add to postfix until '(' is found.
 - Discard the pair of parentheses.
 - If **Operator (+, -, *, /, ^)** →
 - While stack is not empty **and** precedence of top of stack \geq precedence of current operator,
pop from stack and add to postfix.
 - Push the current operator onto the stack.
5. **After reading the entire expression**, pop all remaining operators from the stack and add them to postfix expression.
6. **Display** the postfix expression.
7. **Stop** the program.

کوڈ (Code) :

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

#define MAX 100

class Stack {
    char stack[MAX];
    int top;
public:
```

```

Stack() {
    top = -1;
}

void push(char c) {
    if (top == MAX - 1)
        cout << "Stack is Full \n";
    else
        stack[++top] = c;
}

char pop() {
    if (top == -1) {
        cout << "Stack Underflow \n";
        return '\0';
    }
    return stack[top--];
}

int empty() {
    if(top == -1)
        return 1;
    else
        return 0;
}

char peek() {
    if (top == -1)
        return '\0';
    return stack[top];
}
};

class Expression {
    char infix[MAX];
    char postfix[MAX];

```

```
public:
    void read();
    void inToPost();
    int precedence(char symbol);
    int space(char c);
    void print();
};

void Expression::read() {
    cout << "Enter the infix expression: ";
    cin.getline(infix, MAX);
}

int Expression::space(char c) {
    if(c == ' ' || c == '\t')
        return 1;
    else
        return 0;
}

int Expression::precedence(char symbol) {
    switch (symbol) {
        case '^':
            return 3;
        case '*':
        case '/':
            return 2;
        case '+':
        case '-':
            return 1;
        default:
            return 0;
    }
}

void Expression::inToPost() {
```

```

Stack s;
int j = 0;
char symbol, next;

for (int i = 0; i < strlen(infix); i++) {
    symbol = infix[i];
    if (!space(symbol)) {
        switch (symbol) {
            case '(':
                s.push(symbol);
                break;

            case ')':
                while ((next = s.pop()) != '(')
                    postfix[j++] = next;
                break;

            case '+':
            case '-':
            case '*':
            case '/':
            case '^':
                while (!s.empty() && precedence(s.peek()) >= precedence(symbol))
                    postfix[j++] = s.pop();
                s.push(symbol);
                break;

            default:
                postfix[j++] = symbol;
        }
    }
}

while (!s.empty())
    postfix[j++] = s.pop();
postfix[j] = '\0';
}

```

```

void Expression::print() {
    cout << "\nThe equivalent postfix expression is: \n" << postfix << endl;
}

int main() {
    Expression exp;
    exp.read();
    exp.inToPost();
    exp.print();
    return 0;
}

```

نتیجہ (Output):

```

C:\Users\Rashid Qamar\Docu... X + v
Enter the infix expression: 7+(9-5)*2
The equivalent postfix expression is:
795-2*+

-----
Process exited after 25.61 seconds with return value 0
Press any key to continue . . . |

Enter the infix expression: K + L - M * N + (O^P) * W / U / V * T + Q
The equivalent postfix expression is:
KL+MN*-OP^W*U/V/T**Q+

-----
Process exited after 85.68 seconds with return value 0
Press any key to continue . . . |

```


Program No. 10**مقصد (Aim) :**

Write a C++ Program to Perform Basic Circular Queue operations using a switch case.

1. Insertion
2. Deletion
3. Display
4. Exit

الگورتھم (Algorithm) :

1. **Start** the program.
2. Define a class **Queue** with:
 - An array Q[] of fixed size.
 - Variables: front, rear, and n (size of array).
 - Functions for:
 1. insert() → to add element.
 2. delete() → to remove element.
 3. display() → to show all queue elements.
3. **Initialize Queue**
 - Set front = -1, rear = -1, n = 5.
4. **Insert Operation**
 - If front = 0 & rear == n-1 OR rear = front - 1, print **"Queue is Full"**.
 - Else, if front == -1, set front = 0.
 - Increment rear and place new element at Q[rear].
 - Print success message.
5. **Delete Operation**
 - If rear == -1, print **"Queue is Empty"**.
 - Else, print Q[front] (deleted element).
 - Increment front.
6. **Display Operation**
 - If queue is empty, then print("Queue is Empty").
 - Else, print all elements from front to rear.
7. **In main() function**
 - Create an object of **Queue** class.
 - Display menu with options:
 - 1 → Insertion
 - 2 → Deletion
 - 3 → Display
 - 4 → Exit

- Take Input from user (choice).
- Use switch-case to perform corresponding operation.
- Repeat menu until user chooses **Exit**.

8. **Stop** the program.

کوڈ (Code):

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

#define n 5

class cQueue{
    int Q[n];
    int front, rear;

public:
    cQueue() {
        front = -1, rear = -1;
    }

    void insert() {
        int val;
        if((front == 0 && rear == n-1) || rear = front - 1) {
            cout << "Queue is Full \n";
        }
        else {
            cout << "Enter value to insert: ";
            cin >> val;
            if(front == -1 && rear == -1)
                front = rear = 0;
            else
                rear = (rear+1)%n ;           //For increment
            Q[rear] = val;
            cout << val << " inserted, Successfully \n";
        }
    }

    void Delete() {
        if(front == -1 && rear == -1) {
            cout << "Queue is Empty \n";
        }
        else {
            cout << Q[front] << " deleted, Successfully \n";
        }
    }
}
```

```

        if(front == rear)
            front = rear = -1;
        else
            front = (front + 1)%n;                //For increment
    }
}

void display() {
    if(front == -1 && rear == -1) {
        cout << "Queue is Empty \n";
    }
    else {
        cout << "Queue elements are: \n";
        int i = front;
        while(i != rear) {
            cout << Q[i] << " ";
            i = (i+1)%n;                        //Incrementing
        }
        cout << Q[i] << endl;
    }
}

};

int main() {
    cQueue q;
    int choice;
    while(choice != 4) {
        cout << "\nOptions: " << endl;
        cout << "1. Insertion" << endl;
        cout << "2. Deletion" << endl;
        cout << "3. Display" << endl;
        cout << "4. Exit" << endl;

        cout << "\nEnter any option: ";
        cin >> choice;

        switch(choice) {
            case 1: q.insert(); break;
            case 2: q.Delete(); break;
            case 3: q.display(); break;
            case 4: cout << "Exiting..."; break;
        }
    }
}

```

نتیجہ (Output):

C:\Users\Rashid Qamar\Docu

+

v

Options:

1. Insertion
2. Deletion
3. Display
4. Exit

Enter any option: 1

Enter value to insert: 10

10 inserted, Successfully

Options:

1. Insertion
2. Deletion
3. Display
4. Exit

Enter any option: 1

Enter value to insert: 20

20 inserted, Successfully

Options:

1. Insertion
2. Deletion
3. Display
4. Exit

Enter any option: 1

Enter value to insert: 30

30 inserted, Successfully

Options:

1. Insertion
2. Deletion
3. Display
4. Exit

Enter any option: 1

Enter value to insert: 40

40 inserted, Successfully

Options:

1. Insertion
2. Deletion
3. Display
4. Exit

Enter any option: 1

Enter value to insert: 50

50 inserted, Successfully

Options:

1. Insertion
2. Deletion
3. Display
4. Exit

Enter any option: 1

Queue is Full

Options:

1. Insertion
2. Deletion
3. Display
4. Exit

Enter any option: 2

10 deleted, Successfully

Options:

1. Insertion
2. Deletion
3. Display
4. Exit

Enter any option: 2

20 deleted, Successfully

Options:

1. Insertion
2. Deletion
3. Display
4. Exit

Enter any option: 1

Enter value to insert: 60

60 inserted, Successfully

Options:

1. Insertion
2. Deletion
3. Display
4. Exit

Enter any option: 1

Enter value to insert: 70

70 inserted, Successfully

Options:

1. Insertion
2. Deletion
3. Display
4. Exit

Enter any option: 3

Queue elements are:

30 40 50 60 70

Options:

1. Insertion
2. Deletion
3. Display
4. Exit

Enter any option: 4

Exiting...

Process exited after 283.7 seconds with return value 0

Press any key to continue . . . |

Program No. 00**مقصد (Aim) :**

Write a C++ Program to convert the Infix expression to a Prefix expression using a Stack.

الگورتھم (Algorithm) :

1. **Start** the program.
2. Define a constant MAX for maximum size of stack.
3. Declare an array stack[MAX] and initialize top = -1.
4. Read the infix expression from user.
5. Scan the infix expression **from right to left**, one symbol at a time.
6. For each symbol:
 - **Case 1: Operand**
 - Directly add the operand to prefix expression.
 - **Case 2: Closing Parenthesis ')'**
 - Push it onto the stack.
 - **Case 3: Opening Parenthesis '('**
 - Pop elements from stack and add to prefix until ')' is found.
 - **Case 4: Operator (+, -, * , / , ^)**
 - While stack is not empty **and** precedence of top of stack \geq precedence of current operator \rightarrow
Pop from stack and add to prefix.
 - Push current operator onto stack.
7. After scanning entire infix expression, pop all remaining operators from stack and add to prefix expression.
8. Reverse the prefix expression to get the final result.
9. Display the prefix expression.
10. **End** the program.

کوڈ (Code) :

```
#include<iostream>
#include<cstring>
using namespace std;
#define MAX 50

class Stack {
    char stack[MAX];
    int top;
```

```

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

    void push(char c) {
        if(top == MAX-1)
            cout << "Stack is Full \n";
        else
            stack[++top] = c;
    }

    char pop() {
        if(top == -1) {
            cout << "Stack is Empty \n";
            return '\0';
        }
        else
            return stack[top--];
    }

    int empty() {
        if(top == -1)
            return 1;
        else
            return 0;
    }

    char peek() {
        if(top == -1)
            return '\0';
        else
            return stack[top];
    }
};

class Expression {
    char infix[MAX];
    char prefix[MAX];

public:
    void read();
    void inToPre();
    int precedence(char symbol);
    int space(char c);

```



```

    void print();
};

void Expression::read() {
    cout << "Enter Infix Expression: ";
    cin.getline(infix, MAX);
}

int Expression::precedence(char symbol) {
    switch(symbol) {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        case '^':
            return 3;
        default :
            return 0;
    }
}

void Expression::inToPre() {
    Stack s;
    int j = 0;
    char symbol, next;
    for(int i = strlen(infix) - 1; i>=0; i--) {           //For Reversing the expression
        symbol = infix[i];
        if( !space(symbol) ) {
            switch(symbol) {

                case '(':
                    while((next = s.pop()) != ')')
                        prefix[j++] = next;
                    break;

                case ')':
                    s.push(symbol);
                    break;

                case '+':
                case '-':
                case '*':
                case '/':

```

```

        case '^':
            while(!s.empty() && precedence(s.peek()) > precedence(symbol))
                prefix[j++] = s.pop();
            s.push(symbol);
            break;

        default:
            prefix[j++] = symbol;
    }
}

while( !s.empty())                //For printing stored Stack Element
    prefix[j++] = s.pop();
prefix[j] = '\0';
}

int Expression::space(char c) {
    if(c == ' ' || c == '\t')
        return 1;
    else
        return 0;
}

void Expression::print() {
    cout << "\nThe equivalent Prefix expression is: \n" << strrev(prefix) << endl;
}

int main() {
    Expression exp;
    exp.read();
    exp.inToPre();
    exp.print();
    return 0;
}

```

نتیجہ (Output):

```
C:\Users\Rashid Qamar\Docu  X  +  v
Enter Infix Expression:  J + K - L / M * N
The equivalent Prefix expression is:
-+JK*/LMN
-----
Process exited after 19.46 seconds with return value 0
Press any key to continue . . . |
```

Program No. 11**المقصد (Aim) :**

Write a C++ Program to Perform Double Ended Queue (DEQueue) Operations using Switch Case.

1. Insert at Front
2. Insert at Rear
3. Delete from Front
4. Delete from Rear
5. Display
6. Exit

الگورتهم (Algorithm) :

1. **Start** the program.
2. Define a constant **N** for maximum size of DEQueue.
3. Declare an array **a[N]** and variables front and rear initialized to -1.
4. Display the **menu** of options:
 - 1 → Insert at front
 - 2 → Insert at rear
 - 3 → Delete at front
 - 4 → Delete at rear
 - 5 → Display
 - 6 → Exit
5. Take **choice** input from user.
6. Perform operations according to choice:
 - **Case 1: Insert at Front**
 - If DEQueue is Full → display overflow message.
 - Else if DEQueue is Empty → set front = rear = 0 and insert element.
 - Else → update front = (front - 1 + N) % N and insert element at a[front].
 - **Case 2: Insert at Rear**
 - If DEQueue is Full → display overflow message.
 - Else if DEQueue is Empty → set front = rear = 0 and insert element.
 - Else → update rear = (rear + 1) % N and insert element at a[rear].
 - **Case 3: Delete from Front**
 - If DEQueue is Empty → display underflow message.
 - Else display a[front] and:
 - If front == rear → reset both to -1.
 - Else update front = (front + 1) % N.

- **Case 4: Delete from Rear**

- If DEQueue is Empty → display underflow message.
- Else display a[rear] and:
 - If front == rear → reset both to -1.
 - Else update rear = (rear - 1 + N) % N.

- **Case 5: Display**

- If DEQueue is Empty → show message.
- Else → traverse from front to rear (using circular increment $(i+1)\%N$) and print all elements.

- **Case 6: Exit**

- Stop the program.

7. Repeat steps until choice = 6.

8. **End** the program.

کوڈ (Code):

```
#include<iostream>
using namespace std;
#define N 50

class DEQueue {
    int a[N];
    int front, rear;

public:
    DEQueue();
    void ins_front(int);
    void ins_rear(int);
    void del_front();
    void del_rear();
    void display();
    bool isFull();
    bool isEmpty();
};

DEQueue::DEQueue() {
    front = rear = -1;
}
```

```

bool DEQueue::isEmpty() {
    if(front == -1)
        return true;
    else
        return false;
}

bool DEQueue::isFull() {
    if((front == 0 && rear == N-1) || rear == front - 1 )
        return true;
    else
        return false;
}

void DEQueue::ins_front(int i) {
    if(isFull()) {
        cout << "DEQueue is Full \n";
        return;
    }
    else {
        if(isEmpty()) {
            front = rear = 0;
        }
        else {
            front = (front - 1 + N) % N;           //For Decrement
        }
        a[front] = i;
        cout << a[front] << " Inserted at front \n";
    }
}

void DEQueue::ins_rear(int i) {
    if(isFull()) {
        cout << "DEQueue is Full \n";
        return;
    }
}

```

```

else {
    if(isEmpty()) {
        front = rear = 0;
    }
    else {
        rear = (rear + 1) % N;           //For Increment
    }
    a[rear] = i;
    cout << a[rear] << " Inserted at rear \n";
}
}

void DEQueue::del_front() {
    if(isEmpty()) {
        cout << "DEQueue is Empty \n";
        return;
    }
    else {
        cout << a[front] << " Deleted from front \n";
        if(front == rear) {
            front = rear = -1;
        }
        else {
            front = (front + 1) % N;       //For Increment
        }
    }
}

void DEQueue::del_rear() {
    if(isEmpty()) {
        cout << "DEQueue is Empty \n";
        return;
    }
    else {
        cout << a[rear] << " Deleted from rear \n";
        if(front == rear) {

```

```

        front = rear = -1;
    }
    else {
        rear = (rear- 1 + N) % N;           //For Decrement
    }
}
}

void DEQueue::display() {
    if(isEmpty()) {
        cout << "DEQueue is Empty \n";
    }
    else {
        int i = front;
        cout << "\nDEQueue element: ";
        while(true) {
            cout << a[i] << " ";
            if(i == rear)
                break;
            else
                i = (i + 1) % N;
        }
    }
    cout << endl;
}

int main() {
    int choice, i;
    DEQueue d;
    do {
        cout << "\nOptions: " << endl;
        cout << "1. Insert at front" << endl;
        cout << "2. Insert at rear" << endl;
        cout << "3. delete at front" << endl;
        cout << "4. Delete at rear" << endl;
        cout << "5. Display" << endl;
    }
}

```



```
cout << "6. Exit" << endl;

cout << "\nEnter any option: ";
cin >> choice;

switch(choice) {
    case 1:
        cout << "Enter element to insert: ";
        cin >> i;
        d.ins_front(i); break;

    case 2:
        cout << "Enter element to insert: ";
        cin >> i;
        d.ins_rear(i); break;

    case 3:
        d.del_front(); break;

    case 4:
        d.del_rear(); break;

    case 5:
        d.display(); break;

    case 6:
        cout << "Exiting..."; break;
}
} while(choice != 6);
return 0;
}
```

نتیجہ (Output):



C:\Users\Rashid Qamar\Docu



Options:

1. Insert at front
2. Insert at rear
3. delete at front
4. Delete at rear
5. Display
6. Exit

Enter any option: 1

Enter element to insert: 10

10 Inserted at front

Options:

1. Insert at front
2. Insert at rear
3. delete at front
4. Delete at rear
5. Display
6. Exit

Enter any option: 1

Enter element to insert: 20

20 Inserted at front

Options:

1. Insert at front
2. Insert at rear
3. delete at front
4. Delete at rear
5. Display
6. Exit

Enter any option: 2

Enter element to insert: 30

30 Inserted at rear

Options:

1. Insert at front
2. Insert at rear
3. delete at front
4. Delete at rear
5. Display
6. Exit

Enter any option: 2

Enter element to insert: 40

40 Inserted at rear

Options:

1. Insert at front
2. Insert at rear
3. delete at front
4. Delete at rear
5. Display
6. Exit

Enter any option: 3

20 Deleted from front

Options:

1. Insert at front
2. Insert at rear
3. delete at front
4. Delete at rear
5. Display
6. Exit

Enter any option: 4

40 Deleted from rear

Options:

1. Insert at front
2. Insert at rear
3. delete at front
4. Delete at rear
5. Display
6. Exit

Enter any option: 5

DEQueue element: 10 30

Options:

1. Insert at front
2. Insert at rear
3. delete at front
4. Delete at rear
5. Display
6. Exit

Enter any option: 6

Exiting...

Process exited after 723.4 seconds with return value 0

Press any key to continue . . . |

Program No. 12**مقصد (Aim) :**

Write a C++ Program to Perform Singly Linked List Operations using a switch case.

1. Insertion at Start
2. Insertion at Specified Position
3. Insertion at End
4. Delete from Start
5. Delete from Specified Position
6. Delete from End
7. Update Element
8. Delete Alternate (Starting from second Node)
9. Display (With Position)
10. Exit

الگورتھم (Algorithm) :

1. **Start** the program.
2. Define a class Node with:
 - data (stores value).
 - next (stores address of next node).
3. Initialize head = NULL (empty list).
4. Display menu of operations:
 - 1 → Insert at beginning
 - 2 → Insert at specified position
 - 3 → Insert at end
 - 4 → Delete from beginning
 - 5 → Delete from specified position
 - 6 → Delete from end
 - 7 → Update a node value
 - 8 → Delete alternate nodes
 - 9 → Display list
 - 10 → Exit
5. Take input from user (choice).
6. Perform operation according to choice:
 - **Case 1: Insert at beginning**
 - Create new node.
 - Set new_node->next = head.
 - Set head = new_node.
 - Print success message.

- **Case 2: Insert at specified position**
 - If position = 1 → Call "Insert at beginning".
 - Else traverse list until (pos-1) node.
 - Create new node.
 - Link new node → new_node->next = prev->next.
 - Update link → prev->next = new_node.
 - Print success message.
- **Case 3: Insert at end**
 - Create new node.
 - If list empty → head = new_node.
 - Else traverse till last node and set last->next = new_node.
 - Print success message.
- **Case 4: Delete from beginning**
 - If list empty → print "List is empty".
 - Else store temp = head.
 - Update head = head->next.
 - Free memory of temp.
 - Print deleted element.
- **Case 5: Delete from specified position**
 - If position = 1 → Call "Delete at beginning".
 - Else traverse list till (pos-1) node.
 - Store node to delete → temp = prev->next.
 - Update link → prev->next = temp->next.
 - Free memory of temp.
 - Print deleted element.
- **Case 6: Delete from end**
 - If list empty → print "List is empty".
 - If only one node → delete head and set head = NULL.
 - Else traverse till second-last node.
 - Delete last node and set second_last->next = NULL.
 - Print deleted element.
- **Case 7: Update a node value**
 - Traverse list till given position.
 - Replace node's data with new_data.
 - Print success message.
- **Case 8: Delete alternate nodes**

- Start from head.
- Delete every second node (2nd, 4th, 6th ...).
- Adjust links accordingly.
- Print deleted elements.
- **Case 9: Display**
 - If list empty → print "List is empty".
 - Else traverse from head to NULL printing data(position).
- **Case 10: Exit**
 - Print "Exiting..." and stop program.

7. Repeat steps until choice = 10.

8. **End** the program.

کوڈ (Code):

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

class Node {                                //Create a Node

public:
    int data;
    Node *next;

    Node(int val) {
        data = val;
        next = NULL;
    }
};

void push_start(Node*& head_add, int new_data) {           //Insert at Start
    Node* new_node = new Node(new_data);                 //Allocate node & putting data
    //new_node -> data = new_data;                         //Putting new-node data in new_data
    new_node -> next = head_add;                           //Putting new-node add in head
    head_add = new_node;
    cout << new_data << " inserted, Successfully\n";
}

void push_position(Node*& head, int pos, int new_data) {   //Insert at a Position
    if(pos <= 0) {
```

```

        cout << "Invalid Position" << endl;
        return;
    }
    if(pos == 1) {
        push_start(head, new_data);           //Calling Insert_start
        return;
    }
    Node* temp = head;
    for(int i = 1; temp != NULL && i < pos-1; i++) {           //Traversing
        temp = temp -> next;
    }
    if(temp == NULL) {
        cout << "Position out of Linked list";
        return;
    }
    Node* new_node = new Node(new_data);
    //new_node -> data = new_data;
    new_node -> next = temp -> next;
    temp -> next = new_node;
    cout << new_data << " inserted, Successfully\n";
}

void push_end(Node*& head, int new_data) {           //Insert at End
    Node* new_node = new Node(new_data);
    //new_node -> data = new_data;
    if(head == NULL) {
        head = new_node;
        return;
    }
    Node* temp = head;
    while (temp -> next != NULL) {
        temp = temp -> next;
    }
    temp -> next = new_node;
    cout << new_data << " inserted, Successfully\n";
}

void del_start(Node*& head) {
    if (head == NULL) {

```



```

    cout << "List is empty\n";
    return;
}
Node* temp = head;
cout << temp->data << " deleted, Successfully \n";
head = head->next;
delete temp;
}

void del_position(Node*& head, int pos) {
    if(pos == 1) {
        del_start(head);                                //Calling Del_start
        return;
    }
    Node* prev = head;
    int current_pos = 1;
    while(current_pos < pos-1 && prev != NULL) {
        //Traversing till before the pos
        prev = prev -> next;
        current_pos++;
    }
    if(prev == NULL) {
        cout << "Position out of Linked list";
        return;
    }
    Node* temp = prev -> next;
    cout << temp -> data << " deleted, Successfully \n";
    prev -> next = prev -> next -> next;
    delete temp;
}

void del_end(Node*& head) {
    if (head == NULL) {
        cout << "List is empty\n";
        return;
    }
    if (head->next == NULL) {                                // Only one node
        cout << head->data << " deleted, Successfully \n";
        delete head;
    }
}

```

```

    head = NULL;
    return;
}
Node* second_last = head;
while (second_last->next->next != NULL) {
    second_last = second_last->next;
}
Node* temp = second_last -> next;
cout << temp -> data << " deleted, Successfully \n";
delete temp;
second_last -> next = NULL;
}

void update(Node*& head, int pos, int new_data) {                //Insert at a Position
    if(pos <= 0) {
        cout << "Invalid Position" << endl;
        return;
    }
    Node* temp = head;
    for(int i = 1; temp != NULL && i < pos-1; i++) {            //Traversing
        temp = temp -> next;
    }
    if(temp == NULL) {
        cout << "Position out of Linked list";
        return;
    }
    temp -> data = new_data;
    cout << new_data << " updated, Successfully\n";
}

void del_alterate(Node*& head) {
    if(head == NULL) {
        cout << "List is Empty \n";
        return;
    }
    Node* prev = head;
    Node* temp = NULL;
    while(prev != NULL && prev -> next != NULL) {
        temp = prev -> next;

```

```

        cout << temp -> data << " deleted , ";
        prev -> next = temp -> next;
        prev = prev -> next;
        delete temp;
    }
    cout << endl;
}

void display(Node* head) {
    int count=1;
    Node* temp = head;
    while(temp != NULL) {
        cout << temp -> data << "(" << count << ")" << " -> ";
        temp = temp -> next;
        count++;
    }
    cout << "NULL" << endl;
}

int main() {
    int choice=0;
    int new_data, pos;
    Node* head = NULL;
    while(choice != 10) {
        cout << "\nOptions---" << endl;
        cout << "1. Insert at beginning" << endl;
        cout << "2. Insert at specified position" << endl;
        cout << "3. Insert at end" << endl;
        cout << "4. Delete from beginning" << endl;
        cout << "5. Delete from specified position" << endl;
        cout << "6. Delete from end" << endl;
        cout << "7. Update the List" << endl;
        cout << "8. Delete Alternate" << endl;
        cout << "9. Display" << endl;
        cout << "10. Exit" << endl;

        cout << "\nEnter your choice: ";
        cin >> choice;

        switch(choice) {

```

//For increment

case 1:

```
cout << "Enter new data: ";  
cin >> new_data;  
push_start(head, new_data);  
break;
```

case 2:

```
cout << "Enter position: ";  
cin >> pos;  
cout << "Enter new data: ";  
cin >> new_data;  
push_position(head, pos, new_data);  
break;
```

case 3:

```
cout << "Enter new data: ";  
cin >> new_data;  
push_end(head, new_data);  
break;
```

case 4:

```
del_start(head); break;
```

case 5:

```
cout << "Enter position: ";  
cin >> pos;  
del_position(head, pos); break;
```

case 6:

```
del_end(head); break;
```

case 7:

```
cout << "Enter position: ";  
cin >> pos;  
cout << "Enter new data: ";  
cin >> new_data;  
update(head, pos, new_data); break;
```

case 8:

```

        del_alternate(head); break;

    case 9:
        cout << "Linked List: \n";
        display(head);
        break;

    case 10:
        cout << "Exiting..."; break;

    }
}
return 0;
}

```

نتیجہ (Output):

```

Options---
1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 1
Enter new data: 10
10 inserted, Successfully

Options---
1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 1
Enter new data: 5
5 inserted, Successfully

```

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 9

Linked List:

5(1) -> 10(2) -> 30(3) -> NULL

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 2

Enter position: 3

Enter new data: 20

20 inserted, Successfully

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 9

Linked List:

5(1) -> 10(2) -> 20(3) -> 30(4) -> NULL

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 4
5 deleted, Successfully

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 6
30 deleted, Successfully

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 9
Linked List:
10(1) -> 20(2) -> NULL

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 5

Enter position: 2

20 deleted, Successfully

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 7

Enter position: 1

Enter new data: 5

5 updated, Successfully

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 9

Linked List:

5(1) -> NULL

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 9

Linked List:

5(1) -> 10(2) -> 15(3) -> 20(4) -> 25(5) -> 30(6) -> 35(7) -> 40(8) -> NULL

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 8

10 deleted , 20 deleted , 30 deleted , 40 deleted ,

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 9

Linked List:

5(1) -> 15(2) -> 25(3) -> 35(4) -> NULL

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Delete Alternate
9. Display
10. Exit

Enter your choice: 10

Exiting...

Process exited after 921.6 seconds with return value 0

Press any key to continue . . . |

Program No. 13**مقصد (Aim) :**

Write a C++ Program to Perform Doubly Linked List Operations using a switch case.

1. Insertion at Start
2. Insertion at Specified Position
3. Insertion at End
4. Delete from Start
5. Delete from Specified Position
6. Delete from End
7. Update Element
8. Display
9. Exit

الگورتھم (Algorithm) :

1. **Start** the program.
2. Define a class **Node** with:
 - data (stores value).
 - prev (stores address of previous node).
 - next (stores address of next node).
3. **Initialize:**
head = NULL, tail = NULL (empty list).
4. **Display menu of operations:**
 - 1 → Insert at beginning
 - 2 → Insert at specified position
 - 3 → Insert at end
 - 4 → Delete from beginning
 - 5 → Delete from specified position
 - 6 → Delete from end
 - 7 → Update a node value
 - 8 → Display
 - 9 → Exit
5. Take input from user (**choice**).
6. Perform operation according to choice:

Case 1: Insert at beginning

 - Create a new node.
 - If list is empty → set head = tail = new_node.
 - Else
 - new_node->next = head
 - head->prev = new_node
 - head = new_node.

- Print success message.

Case 2: Insert at specified position

- If position = 1 → Call “Insert at beginning”.
- Else traverse list till (pos-1) node using temporary pointer.
- If position out of range → print “Invalid position”.
- Else
 - new_node->next = temp->next
 - new_node->prev = temp
 - If temp->next != NULL → temp->next->prev = new_node
 - temp->next = new_node.
- If new node inserted at end → update tail = new_node.
- Print success message.

Case 3: Insert at end

- Create a new node.
- If list empty → set head = tail = new_node.
- Else
 - tail->next = new_node
 - new_node->prev = tail
 - tail = new_node.
- Print success message.

Case 4: Delete from beginning

- If list empty → print “List is empty”.
- Else print deleted node data.
- If only one node → set head = tail = NULL.
- Else
 - head = head->next
 - head->prev = NULL.
- Free memory of deleted node.

Case 5: Delete from specified position

- If list empty → print “List is empty”.
- If position = 1 → Call “Delete from beginning”.
- Else traverse list till given position.
- If position out of range → print “Invalid position”.
- Else if deleting last node → Call “Delete from end”.
- Else
 - temp->prev->next = temp->next
 - temp->next->prev = temp->prev.
- Free memory of deleted node.

- Print deleted element.

Case 6: Delete from end

- If list empty → print “List is empty”.
- Else print deleted node data.
- If only one node → set head = tail = NULL.
- Else
 - tail = tail->prev
 - tail->next = NULL.
- Free memory of deleted node.

Case 7: Update a node value

- Traverse list till given position.
- If position out of range → print “Invalid position”.
- Else
 - Replace node's data with new_data.
- Print success message.

Case 8: Display

- If list empty → print “List is empty”.
- Else start from head and traverse using next.
- Print each node's data sequentially till NULL.

Case 9: Exit

- Print “Exiting...” and stop program.

7. Repeat steps **until choice = 10**.

8. **End** the program.

کوڈ (Code):

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

```
class Node{
public:
```

```
    int data;
    Node *prev;
    Node *next;
```

```
    Node(int val) {
        data = val;
        prev = NULL;
```

```

        next = NULL;
    }
};

void ins_start(Node*& head, Node*& tail, int new_data) {
    Node* new_node = new Node(new_data);
    if(head == NULL) {
        head = tail = new_node;
    }
    else {
        new_node -> next = head;
        head -> prev = new_node;
        head = new_node;
    }
    cout << new_data << " inserted at beginning \n";
}

void ins_end(Node*& head, Node*& tail, int new_data) {
    Node* new_node = new Node(new_data);
    if (head == NULL) {
        head = tail = new_node;
    }
    else {
        tail -> next = new_node;
        new_node -> prev = tail;
        tail = new_node;
    }
    cout << new_data << " inserted at end\n";
}

void ins_pos(Node*& head, Node*& tail, int pos, int new_data) {
    if(pos <= 0) {
        cout << "Invalid Position \n";
        return;
    }
    if(pos == 1) {

```

```

        ins_start(head, tail, new_data);
        return;
    }
    Node* temp = head;
    for(int i = 1; temp != NULL && i < pos-1; i++) {
        temp = temp -> next;
    }
    if(temp == NULL || temp == tail) {
        ins_end(head, tail, new_data);
        return;
    }
    Node* new_node = new Node(new_data);
    new_node -> next = temp -> next;
    new_node -> prev = temp;
    temp -> next -> prev = new_node;
    temp -> next = new_node;
    cout << new_data << " inserted at position " << pos << "\n";
}

void del_start(Node*& head, Node*& tail) {
    if(head == NULL) {
        cout << "List in Empty \n";
        return;
    }
    Node* temp = head;
    cout << temp -> data << " deleted from beginning \n";
    if(head == tail) {
        head = tail = NULL;
    }
    else {
        head = head -> next;
        head -> prev = NULL;
    }
    delete temp;
}

```

```

void del_end(Node*& head, Node*& tail) {
    if(tail == NULL) {
        cout << "List is Empty \n";
        return;
    }
    Node* temp = tail;
    cout << temp -> data << " deleted from end \n";
    if(head == tail) {
        head = tail = NULL;
    }
    else {
        tail = tail -> prev;
        tail -> next = NULL;
    }
    delete temp;
}

```

```

void del_pos(Node*& head, Node*& tail, int pos) {
    if(head == NULL) {
        cout << "List is Empty \n";
        return;
    }
    if(pos == 1) {
        del_start(head, tail);
    }
    Node* temp = head;
    for(int i = 1; temp != NULL && i < pos-1; i++) {
        temp = temp -> next;
    }
    if(temp == NULL) {
        cout << "Position out of range \n";
        return;
    }
    if(temp == tail) {
        del_end(head, tail);
        return;
    }
}

```



```

    }
    cout << temp -> data << " deleted from position " << pos << "\n";
    temp -> prev -> next = temp -> next;
    temp -> next -> prev = temp -> prev;
    delete temp;
}

```

```

void update(Node*& head, int pos, int new_data) {
    if(head == NULL) {
        cout << "List is Empty \n";
        return;
    }
    Node* temp = head;
    for(int i=1; temp != NULL && i<pos; i++) {
        temp = temp -> next;
    }
    if(temp == NULL) {
        cout << "Position out of Range \n";
        return;
    }
    temp -> data = new_data;
    cout << "Node Updated at position " << pos << "\n";
}

```

```

void display(Node* head) {
    if(head == NULL) {
        cout << "List is Empty \n";
        return;
    }
    cout << "List: \n";
    Node* temp = head;
    int count = 1;
    while (temp != NULL) {
        cout << temp -> data << "(" << count << ") -> ";
        temp = temp -> next;
        count++;
    }
}

```

```

    }
    cout << "NULL \n";
}

int main() {
    Node* head = NULL;
    Node* tail = NULL;
    int choice=0, new_data, pos;

    while (choice != 9) {
        cout << "\nOptions---" << endl;
        cout << "1. Insert at beginning" << endl;
        cout << "2. Insert at specified position" << endl;
        cout << "3. Insert at end" << endl;
        cout << "4. Delete from beginning" << endl;
        cout << "5. Delete from specified position" << endl;
        cout << "6. Delete from end" << endl;
        cout << "7. Update the List" << endl;
        cout << "8. Display" << endl;
        cout << "9. Exit" << endl;

        cout << "\nEnter your choice: ";
        cin >> choice;

        switch(choice) {
            case 1:
                cout << "Enter new data: ";
                cin >> new_data;
                ins_start(head, tail, new_data);
                break;

            case 2:
                cout << "Enter position: ";
                cin >> pos;
                cout << "Enter new data: ";
                cin >> new_data;

```

```
    ins_pos(head, tail, pos, new_data);  
    break;
```

case 3:

```
    cout << "Enter new data: ";  
    cin >> new_data;  
    ins_end(head, tail, new_data);  
    break;
```

case 4:

```
    del_start(head, tail);  
    break;
```

case 5:

```
    cout << "Enter position: ";  
    cin >> pos;  
    del_pos(head, tail, pos);  
    break;
```

case 6:

```
    del_end(head, tail);  
    break;
```

case 7:

```
    cout << "Enter position: ";  
    cin >> pos;  
    cout << "Enter new data: ";  
    cin >> new_data;  
    update(head, pos, new_data);  
    break;
```

case 8:

```
    display(head);  
    break;
```

case 9:

```

        cout << "Exiting..."; break;
    }
}
return 0;
}

```

نتیجہ (Output):

```

Options---
1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Display
9. Exit

Enter your choice: 1
Enter new data: 10
10 inserted at beggining

Options---
1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Display
9. Exit

Enter your choice: 3
Enter new data: 30
30 inserted at end

```

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Display
9. Exit

Enter your choice: 8

List:

10(1) -> 30(2) -> NULL

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Display
9. Exit

Enter your choice: 2

Enter position: 2

Enter new data: 20

20 inserted at position 2

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Display
9. Exit

Enter your choice: 8

List:

10(1) -> 20(2) -> 30(3) -> NULL

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Display
9. Exit

Enter your choice: 4
10 deleted from beginning

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Display
9. Exit

Enter your choice: 6
30 deleted from end

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Display
9. Exit

Enter your choice: 8
List:
20(1) -> NULL

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Display
9. Exit

Enter your choice: 7

Enter position: 1

Enter new data: 25

Node Updated at position 1

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Display
9. Exit

Enter your choice: 8

List:

25(1) -> NULL

Options---

1. Insert at beginning
2. Insert at specified position
3. Insert at end
4. Delete from beginning
5. Delete from specified position
6. Delete from end
7. Update the List
8. Display
9. Exit

Enter your choice: 9

Exiting...

Process exited after 589.7 seconds with return value 0

Press any key to continue . . . |

Program No. 14**مقصد (Aim) :**

Write a C++ Program to Perform Bubble Sort Technique.

الگورتھم (Algorithm) :

1. **Start** the program.
2. Declare an array **arr[]** and integer variable **n** for size.
3. **Input** the number of elements **n** and then input **n** elements into **arr[]**.
4. Create the function **bubble_sort(arr, n)** to sort the array.
5. In **bubble_sort** function
 - Repeat for **i = 0** to **n - 2**:
 - Set **swapped = false**.
 - Repeat for **j = 0** to **n - 2 - i**:
 - If **arr[j] > arr[j + 1]**, then
Swap **arr[j]** and **arr[j + 1]**.
 - Set **swapped = true**.
 - If **swapped == false**, break (array already sorted).
6. Return to main program.
7. **Display** the sorted array.
8. **End** the program.

کوڈ (Code) :

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

void bubble_sort(int arr[], int n) {
    int temp;
    bool swapped;

    for(int i=0; i<n-1; i++) {
        swapped = false;

        for(int j=0; j<n-1-i; j++) {
            if(arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
```



```

        swapped = true;
    }
}

    if(!swapped)                                //If Already swapped
        break;
}
}

int main() {
    int n;
    cout << "Enter number of elements: ";
    cin >> n;

    int arr[n];
    cout << "Enter " << n << " elements: \n";
    for(int i=0; i<n; i++)
        cin >> arr[i];

    bubble_sort(arr, n);

    cout << "\nSorted list: \n";
    for(int i=0; i<n; i++) {
        cout << arr[i] << "\t";
    }
    cout << endl;

    return 0;
}

```

نتیجہ (Output):

```
C:\Users\Rashid Qamar\Docu  ×  +  v
Enter number of elements: 5
Enter 5 elements:
81 79 86 33 66

Sorted list:
33      66      79      81      86

-----
Process exited after 12.65 seconds with return value 0
Press any key to continue . . . |
```

Program No. 15**مقصد (Aim) :**

Write a C++ Program to Perform Insertion Sort Technique.

الگورتھم (Algorithm) :

1. **Start** the program.
2. **Declare** an array arr[] and a variable n for number of elements.
3. **Take input** from user for n and then the array elements.
4. **Create a function** insertion_sort(arr, n).
5. Inside insertion_sort:
 - Repeat for i = 1 to n-1:
 - Set temp = arr[i]
 - Set j = i - 1
 - While j >= 0 and arr[j] > temp
 - Move element one position ahead → arr[j + 1] = arr[j]
 - Decrement j
 - Place temp at correct position → arr[j + 1] = temp
6. **Return to main**, display the sorted array.
7. **End** the program.

کوڈ (Code) :

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

void insertion_sort(int arr[], int n) {
    int i, j, temp;

    for(int i=1; i<n; i++) {
        temp = arr[i];
        j = i - 1;

        while (j >= 0 && arr[j] > temp) {
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = temp;
    }
}
```

```

    }
}

int main() {

    int n;
    cout << "Enter number of elements: ";
    cin >> n;

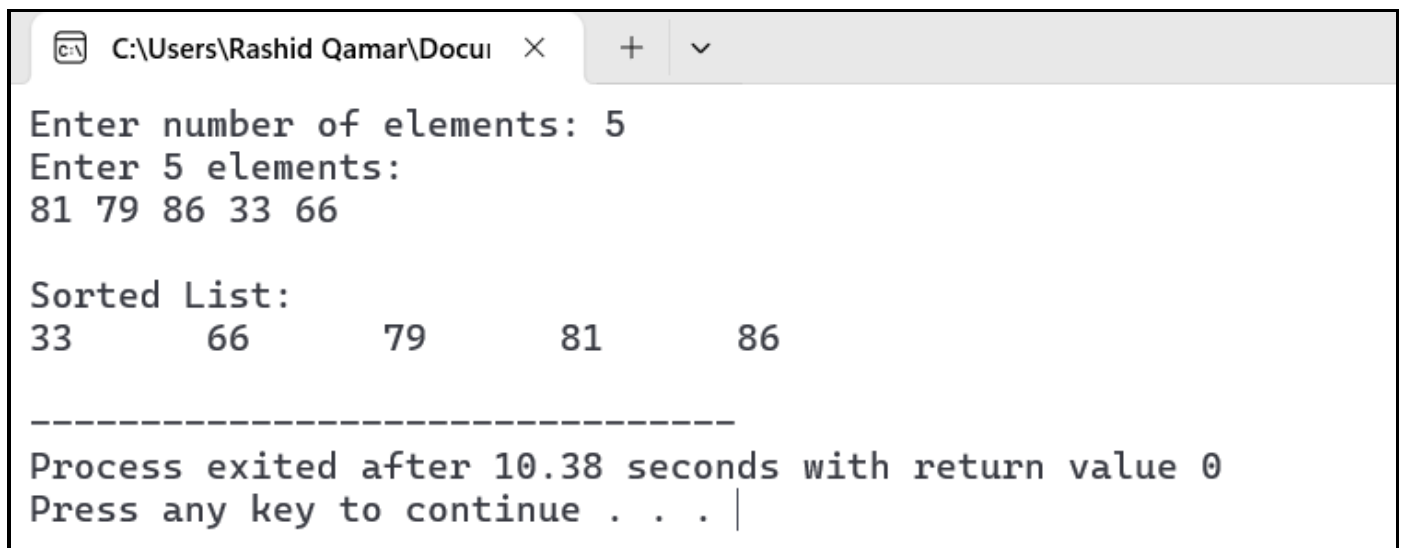
    int arr[n];
    cout << "Enter " << n << " elements: \n";
    for(int i=0; i<n; i++) {
        cin >> arr[i];
    }

    insertion_sort(arr, n);

    cout << "\nSorted List: \n";
    for(int i=0; i<n; i++) {
        cout << arr[i] << "\t";
    }
    cout << endl;
    return 0;
}

```

نتیجہ (Output):



```

C:\Users\Rashid Qamar\Docu
Enter number of elements: 5
Enter 5 elements:
81 79 86 33 66

Sorted List:
33      66      79      81      86

-----
Process exited after 10.38 seconds with return value 0
Press any key to continue . . . |

```

Program No. 16**مقصد (Aim) :**

Write a C++ Program to Perform Merge Sort Technique.

الگورتھم (Algorithm) :

1. **Start** the program.
2. **Declare** an array arr[] and integer n for number of elements.
3. **Input** the value of n and the elements of the array.
4. **Call** the function split(arr, 0, n-1).
5. Inside split:
 - If low < high:
 1. Find mid = (low + high) / 2
 2. Recursively call mergeSort(arr, low, mid)
 3. Recursively call mergeSort(arr, mid + 1, high)
 4. Call merge(arr, low, mid, high) to combine the two sorted halves
6. Inside merge_sort:
 - Compare elements of both halves and store the smaller one into a temporary array
 - Copy the remaining elements (if any) from both halves
 - Copy all elements back into the original array
7. **Display** the sorted array.
8. **End** the program.

کوڈ (Code) :

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

void merge_sort(int arr[], int low, int high, int mid) {
    int i, j, k, b[50];
    i = low;
    k = low;
    j = mid + 1;
    while (i <= mid && j <= high) {
        if(arr[i] < arr[j]) {
            b[k] = arr[i];
            i++;
        }
    }
}
```

```

        }
        else {
            b[k] = arr[j];
            j++;
        }
        k++;
    }
    if(i > mid) {
        while (j <= high) {
            b[k] = arr[j];
            j++;
            k++;
        }
    }
    else {
        while(i <= mid) {
            b[k] = arr[i];
            i++;
            k++;
        }
    }
    for(int m=low; m<k; m++) {
        arr[m] = b[m];
    }
}

void split(int arr[], int low, int high) {
    int mid;
    if(low < high) {
        mid = (low+high)/2;
        split(arr, low, mid);      //For left half
        split(arr, mid+1, high);   //For Right Half
        merge_sort(arr, low, high, mid); //For merging
    }
}

```

```

int main() {
    int array[50], num;
    cout << "Enter number of elements: ";
    cin >> num;
    cout << "Enter " << num << " elements \n";
    for(int i=0; i<num; i++) {
        cin >> array[i];
    }
    split(array, 0, num-1);
    cout << "\nSorted Array: \n";
    for(int i=0; i<num; i++) {
        cout << array[i] << "\t";
    }
    return 0;
}

```

نتیجہ (Output):

```

C:\Users\Rashid Qamar\Docu
Enter number of elements: 5
Enter 5 elements
81 79 86 33 66

Sorted Array:
33      66      79      81      86
-----
Process exited after 9.194 seconds with return value 0
Press any key to continue . . .

```

Program No. 17**مقصد (Aim) :**

Write a C++ Program to Perform Selection Sort Technique.

الگورتھم (Algorithm) :

1. **Start** the program.
2. **Declare** an array arr[] to store elements.
3. **Input** the total number of elements n.
4. **Read** n elements from the user and store them in arr[].
5. **Define a function** find_smallest(arr, i, n) which:
 - Takes the array and starting index i.
 - Initializes ele_small = arr[i] and position = i.
 - Compares each element from i+1 to n-1.
 - If a smaller element is found, update ele_small and position.
 - Returns the index of the smallest element (position).
6. **In the main function**, repeat for each element i from 0 to n-2:
 - Call pos = findSmallest(arr, i, n).
 - Swap arr[i] with arr[pos].
7. **Display** the sorted array after all passes.
8. **Stop** the program.

کوڈ (Code) :

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

int find_smallest(int arr[], int start, int n) {
    int pos = start;
    for(int j = start+1; j < n; j++) {
        if(arr[j] < arr[pos]) {
            pos = j;
        }
    }
    return pos;
}

void selection_sort(int arr[], int n) {
    int pos;
```



```

    for(int i=0; i<n-1; i++) {
        pos = find_smallest(arr, i, n);
        int temp = arr[i];
        arr[i] = arr[pos];
        arr[pos] = temp;
    }
}

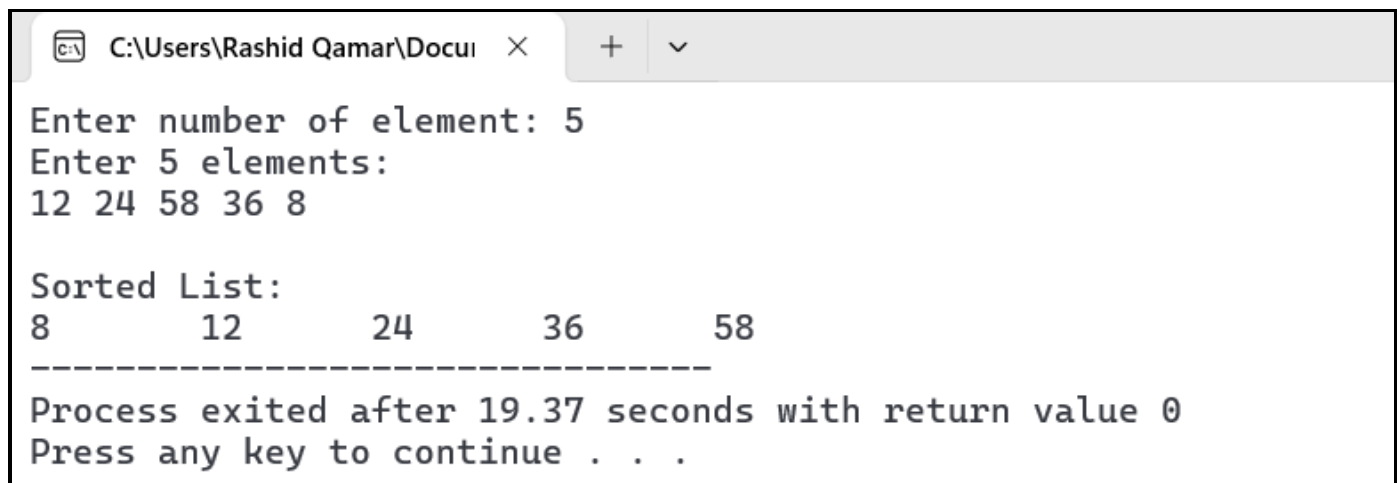
int main() {
    int n;
    cout << "Enter number of element: ";
    cin >> n;

    int arr[n];
    cout << "Enter " << n << " elements: \n";
    for(int i=0; i<n; i++) {
        cin >> arr[i];
    }
    selection_sort(arr, n);                                //Calling Function

    cout << "\nSorted List: \n";
    for(int i=0; i<n; i++) {
        cout << arr[i] << "\t";
    }
    return 0;
}

```

نتیجہ (Output):



```

C:\Users\Rashid Qamar\Docu  x  +  v
Enter number of element: 5
Enter 5 elements:
12 24 58 36 8

Sorted List:
8      12      24      36      58
-----
Process exited after 19.37 seconds with return value 0
Press any key to continue . . .

```

Program No. 18**مقصد (Aim) :**

Write a C++ Program to Perform Quick Sort Technique.

الگورتھم (Algorithm) :

1. **Start** the program.
2. **Input** the total number of elements n.
3. **Read** n elements into array arr[].
4. Define a function **swap(a, b)**
Exchanges the values of two elements.
5. Define a function **partition(arr, low, high)**
 - Choose arr[high] as the **pivot**.
 - Initialize i = low - 1.
 - For each element j from low to high - 1:
 - If $\text{arr}[j] \leq \text{pivot}$, increment i and swap arr[i] with arr[j].
 - Swap arr[i+1] with the pivot.
 - Return i + 1 as the **pivot position**.
6. Define recursive function **quickSort(arr, low, high)**
 - If low < high:
 - Call partition() to find pivot index p.
 - Recursively sort left subarray (low to p-1).
 - Recursively sort right subarray (p+1 to high).
7. **Display** the sorted array.
8. **End** the program.

کوڈ (Code) :

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

void swap_num(int &a, int &b) {                               //For swapping number
    int temp = a;
    a = b;
    b = temp;
}

int partition(int arr[], int low, int high) {
```

```

    int pivot = arr[high];
    int i = low - 1;

    for(int j=low; j<high; j++) {
        if(arr[j] <= pivot) {
            i++;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i+1], arr[high]);
    return (i+1);
}

void quick_sort(int arr[], int low, int high) {
    if(low < high) {
        int pivotIndex = partition(arr, low, high);
        quick_sort(arr, low, pivotIndex-1);
        quick_sort(arr, pivotIndex+1, high);
    }
}

int main() {
    int n;
    cout << "Enter number of elements: ";
    cin >> n;

    int arr[n];
    cout << "Enter " << n << " elements: \n";
    for(int i=0; i<n; i++) {
        cin >> arr[i];
    }

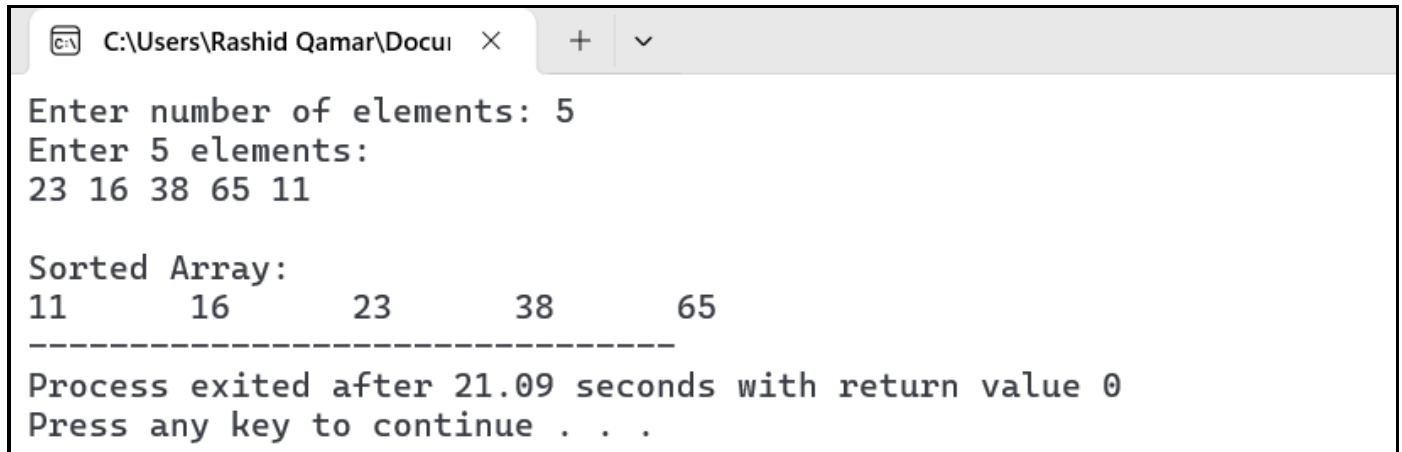
    quick_sort(arr, 0, n-1);

    cout << "\nSorted Array: \n";
    for(int i=0; i<n; i++) {

```

```
        cout << arr[i] << "\\t";  
    }  
  
    return 0;  
}
```

نتیجہ (Output):



```
C:\Users\Rashid Qamar\Docu >  
Enter number of elements: 5  
Enter 5 elements:  
23 16 38 65 11  
  
Sorted Array:  
11      16      23      38      65  
-----  
Process exited after 21.09 seconds with return value 0  
Press any key to continue . . .
```