LAB MANUALS

Name：Muhammad Masood

Roll no：2023-bs-ai-056

Submitted to：Mam Irsha Qureshi

Degree：AI - SEC (A)

LAB 01

Array

Arrays are a collection of elements, all of the same type, stored in contiguous memory locations.

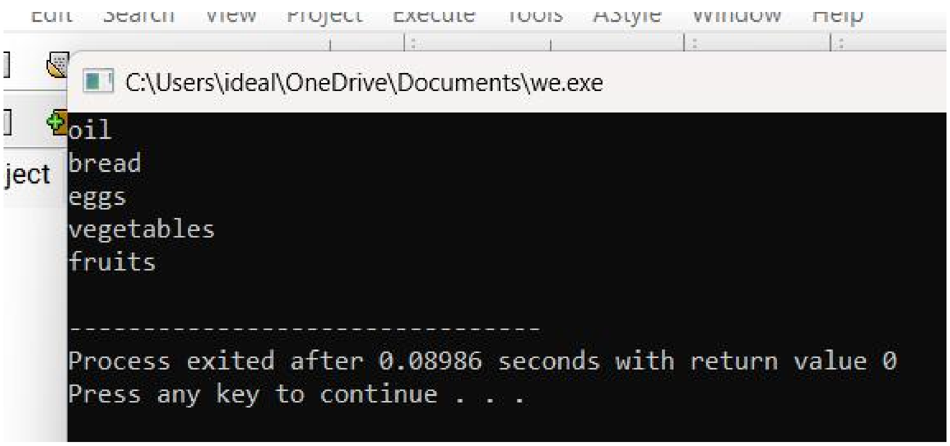
Syntax:

type arrayName[size];

code 1

```
#include<iostream>
 using namespace std;
 int main()
{ string groceryItems[5] = {"oil", "bread", "eggs", "vegetables", "fruits"};
 for (int index = 0; index < 5; index++)
{ cout << groceryItems[index] << endl;
 }
return 0;
}
```

Output:



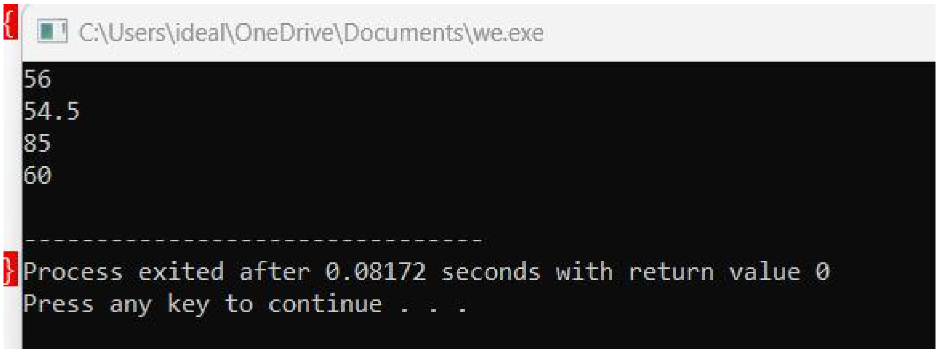
```
oil
bread
eggs
vegetables
fruits

--------------------------------
Process exited after 0.08986 seconds with return value 0
Press any key to continue . . .
```

:

Code 2:

```
#include<iostream>
using namespace std;
int main()
{
   float arr[4]={56 , 54.5 , 85 , 60 };
   for(int i=0 ; i<4; i++)
   {
      cout<<arr[i]<<endl;
   }
   return 0;
}
```

Output:

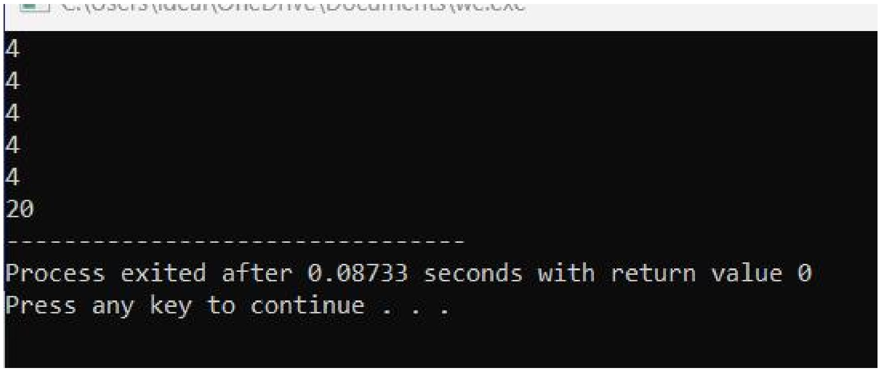


```
56
54.5
85
60

--------------------------------
Process exited after 0.08172 seconds with return value 0
Press any key to continue . . .
```

Code 3:

```
#include<iostream>
using namespace std;
int main()
{
   float arr[5]={ 54.5 , 60 , 50.5 , 66 , 25.5 };
   for(int i=0 ; i<5; i++)
   {
      cout<<sizeof(arr[i])<<endl;
   }
   cout<<sizeof(arr);
   return 0;
}
```

Output:



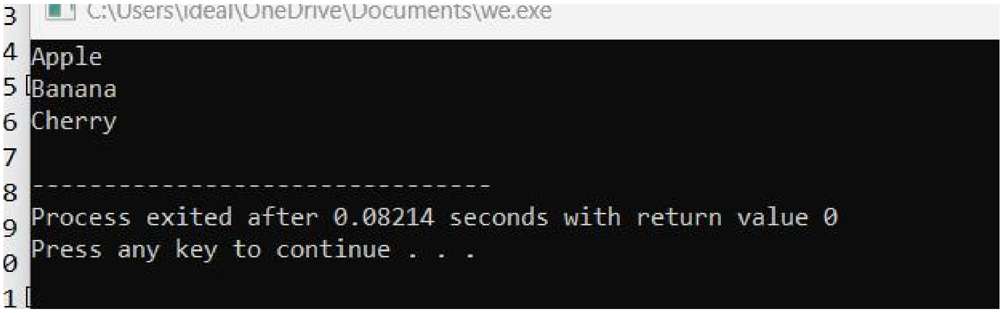
```
4
4
4
4
4
20
--------------------------------
Process exited after 0.08733 seconds with return value 0
Press any key to continue . . .
```

Code 4:

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

int main() {
   string fruits[3];
   fruits[0] = "Apple";
   fruits[1] = "Banana";
   fruits[2] = "Cherry";
```

```
    for(int i = 0; i < 3; i++) {
        cout << fruits[i] << "\n";
    }
```

Output:

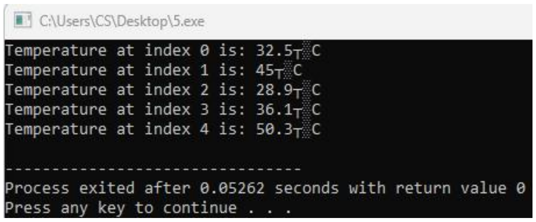


```
C:\Users\ideal\OneDrive\Documents\we.exe
Apple
Banana
Cherry

--------------------------------
Process exited after 0.08214 seconds with return value 0
Press any key to continue . . .
```

Code 5:

```
#include <iostream>
using namespace std;
int main()
{
    double temperatures[5] = {32.5, 45.0, 28.9, 36.1, 50.3};
    for (int index = 0; index < 5; index++) {
        cout << "Temperature at index " << index << " is: " << temperatures[index] <<
"°C" << endl;
    }
    return 0;}
```

output:



```
C:\Users\CS\Desktop\5.exe
Temperature at index 0 is: 32.5┬░C
Temperature at index 1 is: 45┬░C
Temperature at index 2 is: 28.9┬░C
Temperature at index 3 is: 36.1┬░C
Temperature at index 4 is: 50.3┬░C

--------------------------------
Process exited after 0.05262 seconds with return value 0
Press any key to continue . . .
```

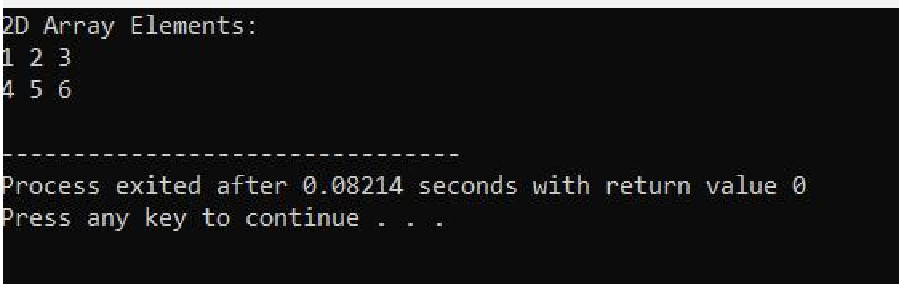
LAB 02

Multi-Dimensional array

Code 1

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

int main() {
    int matrix[2][3] = {
        {1, 2, 3},
        {4, 5, 6}
    };

    cout << "2D Array Elements:\n";
    for (int row = 0; row < 2; row++) {
        for (int col = 0; col < 3; col++) {
            cout << matrix[row][col] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Output :

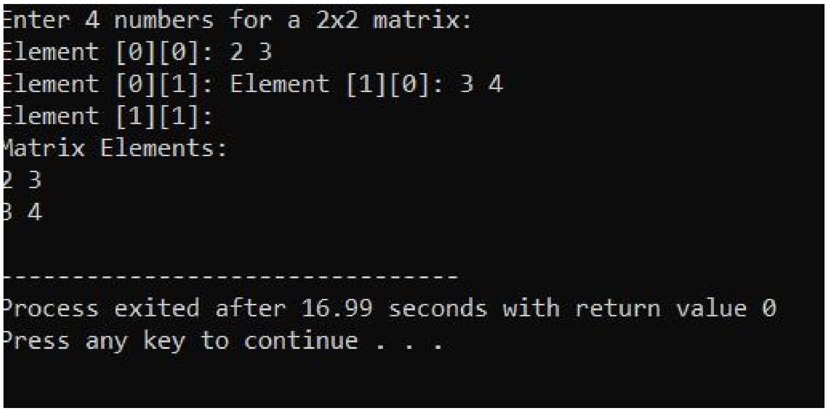


```
2D Array Elements:
1 2 3
4 5 6

--------------------------------
Process exited after 0.08214 seconds with return value 0
Press any key to continue . . .
```

Code 2:

```
#include <iostream>

using namespace std;


int main() {

    int matrix[2][2];

    cout << "Enter 4 numbers for a 2x2 matrix:\n";


    for (int row = 0; row < 2; row++) {

        for (int col = 0; col < 2; col++) {

            cout << "Element [" << row << "][" << col << "]: ";

            cin >> matrix[row][col];

        }

    }


    cout << "\nMatrix Elements:\n";

    for (int row = 0; row < 2; row++) {

        for (int col = 0; col < 2; col++) {
            cout << matrix[row][col] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Output :

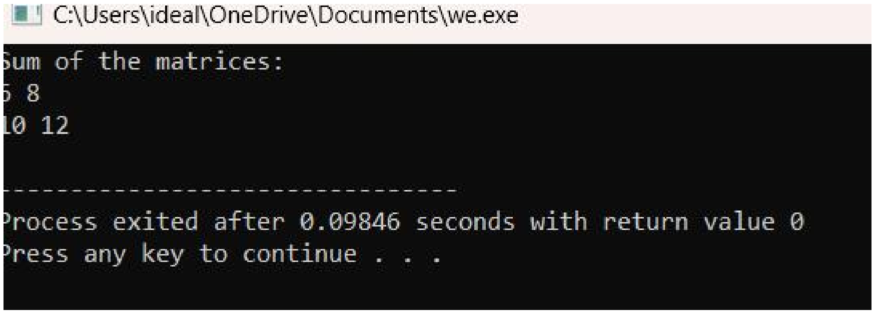


```
Enter 4 numbers for a 2x2 matrix:
Element [0][0]: 2 3
Element [0][1]: Element [1][0]: 3 4
Element [1][1]:
Matrix Elements:
2 3
3 4

--------------------------------
Process exited after 16.99 seconds with return value 0
Press any key to continue . . .
```

Code 3:

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

int main() {
    int matrix1[2][2] = {{1, 2}, {3, 4}};
    int matrix2[2][2] = {{5, 6}, {7, 8}};
    int sum[2][2];
    for (int row = 0; row < 2; row++) {
        for (int col = 0; col < 2; col++) {
            sum[row][col] = matrix1[row][col] + matrix2[row][col];
        }
    }

    cout << "Sum of the matrices:\n";
    for (int row = 0; row < 2; row++) {
        for (int col = 0; col < 2; col++) {
            cout << sum[row][col] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Output:

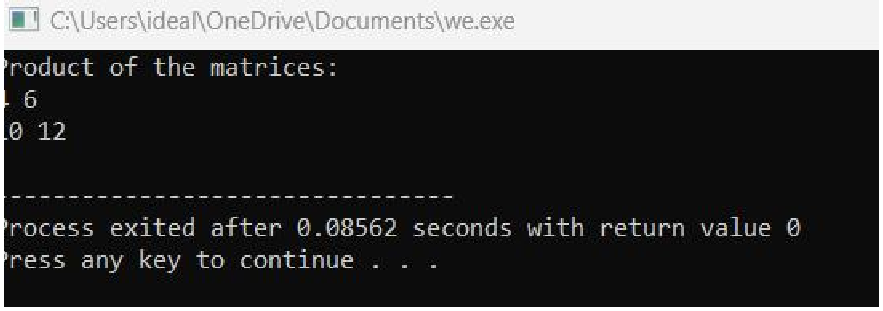


```
Sum of the matrices:
5 8
10 12

--------------------------------
Process exited after 0.09846 seconds with return value 0
Press any key to continue . . .
```

Code 4:

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

int main() {
    int matrix1[2][2] = {{1, 2}, {3, 4}};
    int matrix2[2][2] = {{2, 0}, {1, 3}};
    int product[2][2] = {{0, 0}, {0, 0}};

    for (int row = 0; row < 2; row++) {
        for (int col = 0; col < 2; col++) {
            for (int k = 0; k < 2; k++) {
                product[row][col] += matrix1[row][k] * matrix2[k][col];
            }
        }
    }

    cout << "Product of the matrices:\n";
    for (int row = 0; row < 2; row++) {
        for (int col = 0; col < 2; col++) {
            cout << product[row][col] << " ";
        }
        cout << endl;
    }
    return 0;

}
```

Output;



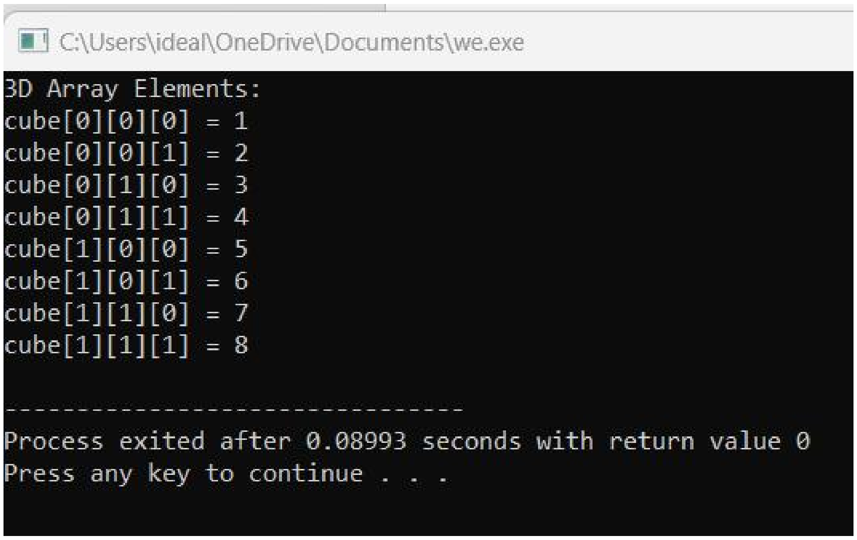
C:\Users\ideal\OneDrive\Documents\we.exe

```
Product of the matrices:
4 6
10 12

--------------------------------
Process exited after 0.08562 seconds with return value 0
Press any key to continue . . .
```

Code 5:

```
#include <iostream>

using namespace std;


int main() {

    int cube[2][2][2] = {

        {{1, 2}, {3, 4}},

        {{5, 6}, {7, 8}}

    };


    cout << "3D Array Elements:\n";

    for (int i = 0; i < 2; i++) {

        for (int j = 0; j < 2; j++) {

            for (int k = 0; k < 2; k++) {
                cout << "cube[" << i << "][" << j << "][" << k << "] = " << cube[i][j][k] << 
endl;
            }
        }
    }


    return 0;
}
```

Output:



```
3D Array Elements:
cube[0][0][0] = 1
cube[0][0][1] = 2
cube[0][1][0] = 3
cube[0][1][1] = 4
cube[1][0][0] = 5
cube[1][0][1] = 6
cube[1][1][0] = 7
cube[1][1][1] = 8

--------------------------------
Process exited after 0.08993 seconds with return value 0
Press any key to continue . . .
```

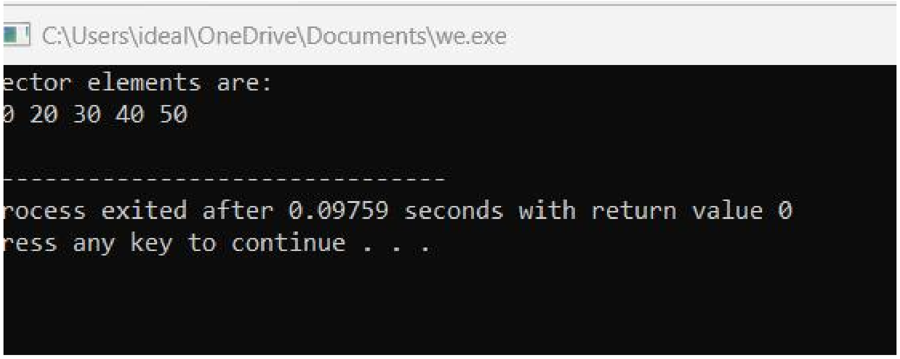
Lab 03

Vectors

code 1:

```
#include <iostream>

#include <vector>

using namespace std;


int main() {

   vector<int> numbers = {10, 20, 30, 40, 50};


   cout << "Vector elements are:\n";

   for (int num : numbers) {

      cout << num << " ";

   }

   cout << endl;


   return 0;

}
```

Output:

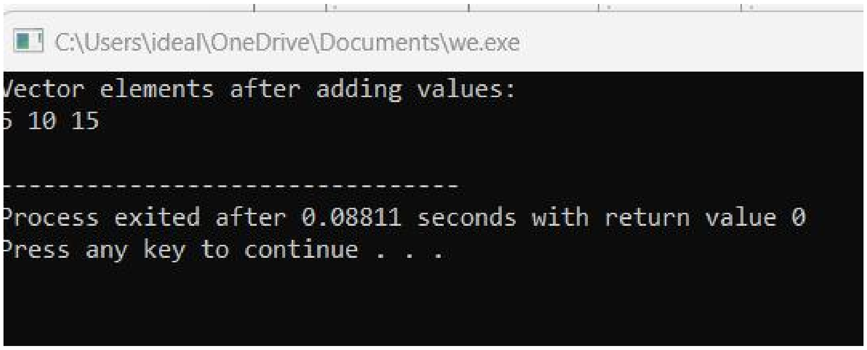


C:\Users\ideal\OneDrive\Documents\we.exe

```
ector elements are:
0 20 30 40 50

--------------------------------
rocess exited after 0.09759 seconds with return value 0
ress any key to continue . . .
```

Code 2:

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

int main() {
    vector<int> numbers;

    // Adding elements to the vector
    numbers.push_back(5);
    numbers.push_back(10);
    numbers.push_back(15);

    cout << "Vector elements after adding values:\n";
    for (int num : numbers) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```



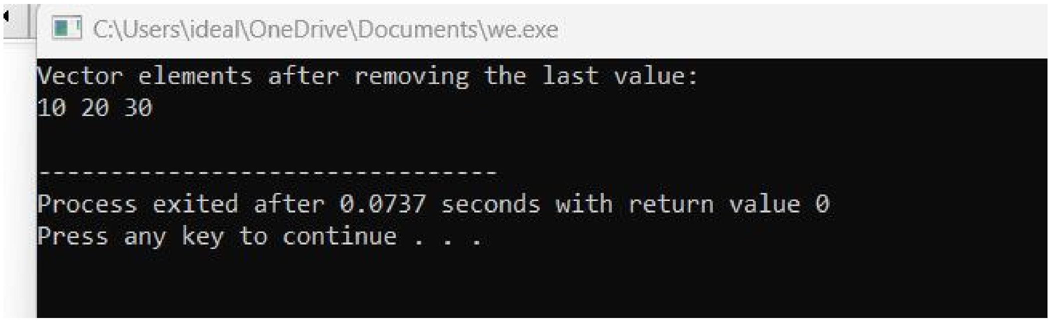
C:\Users\ideal\OneDrive\Documents\we.exe

```
Vector elements after adding values:
5 10 15

--------------------------------
Process exited after 0.08811 seconds with return value 0
Press any key to continue . . .
```

Code 3:

```
#include <iostream>

#include <vector>

using namespace std;


int main() {

   vector<int> numbers = {10, 20, 30, 40};


   // Removing the last element

   numbers.pop_back();


   cout << "Vector elements after removing the last value:\n";

   for (int num : numbers) {

      cout << num << " ";

   }

   cout << endl;


   return 0;
}
```

Output:

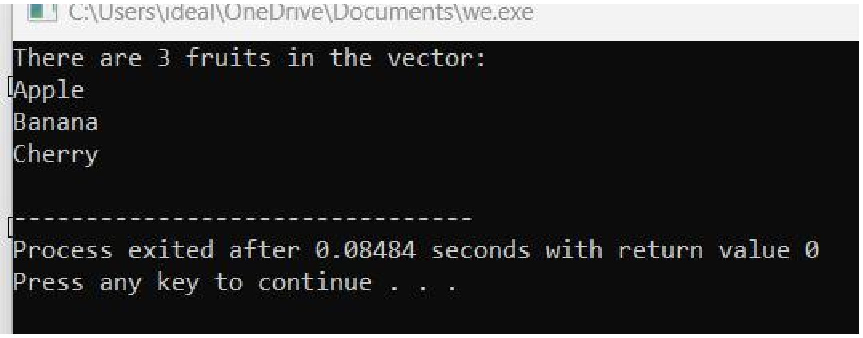


```
Vector elements after removing the last value:
10 20 30

--------------------------------
Process exited after 0.0737 seconds with return value 0
Press any key to continue . . .
```

Code 4:

```
#include <iostream>

#include <vector>

using namespace std;


int main() {

   vector<string> fruits = {"Apple", "Banana", "Cherry"};


   cout << "There are " << fruits.size() << " fruits in the vector:\n";

   for (size_t i = 0; i < fruits.size(); i++) {

      cout << fruits[i] << endl;

   }


   return 0;

}
```

Output:

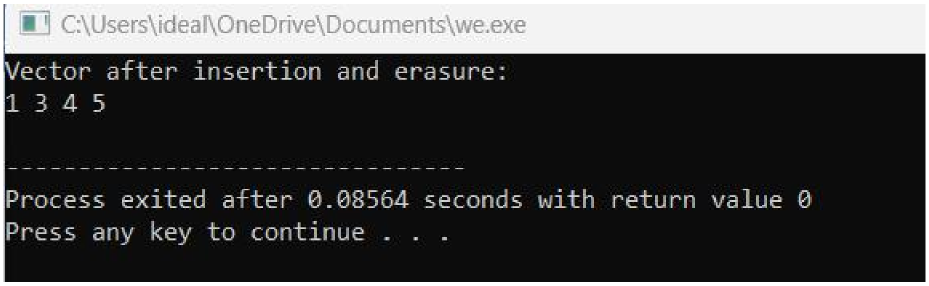


```
C:\Users\ideal\OneDrive\Documents\we.exe
There are 3 fruits in the vector:
Apple
Banana
Cherry

--------------------------------
Process exited after 0.08484 seconds with return value 0
Press any key to continue . . .
```

Code 5:

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

int main() {
    vector<int> numbers = {1, 2, 4, 5};

    // Insert 3 at the 3rd position (index 2)
    numbers.insert(numbers.begin() + 2, 3);

    // Erase the 2nd element (index 1)
    numbers.erase(numbers.begin() + 1);

    cout << "Vector after insertion and erasure:\n";
    for (int num : numbers) {
        cout << num << " ";
    }
    cout << endl;


    return 0;
}
```

Output:



```
Vector after insertion and erasure:
1 3 4 5

--------------------------------
Process exited after 0.08564 seconds with return value 0
Press any key to continue . . .
```

LAB 04

List

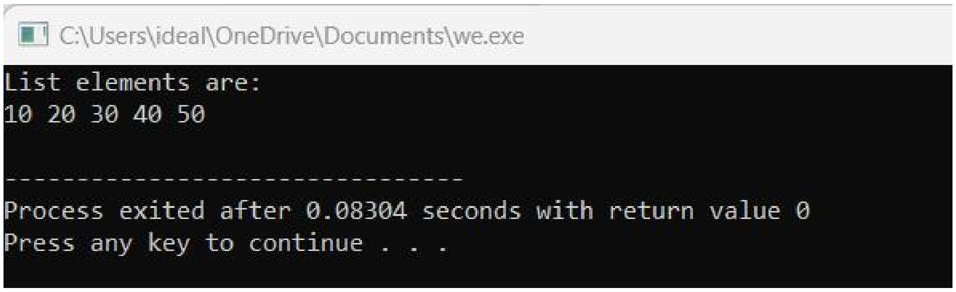
Syntax:

```
#include <list>
std::list<type> list_name;
```

code 1:

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

int main() {
    list<int> numbers = {10, 20, 30, 40, 50};

    cout << "List elements are:\n";
    for (int num : numbers) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

Output:



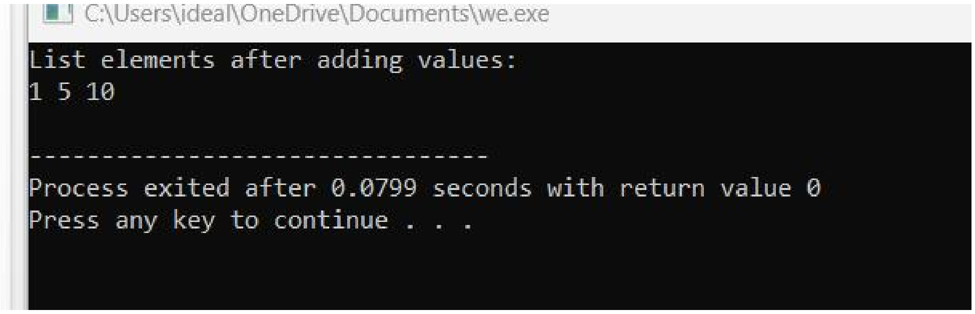
```
List elements are:
10 20 30 40 50

--------------------------------
Process exited after 0.08304 seconds with return value 0
Press any key to continue . . .
```

Code 2:

```
#include <iostream>

#include <list>

using namespace std;


int main() {

   list<int> numbers;


   // Adding elements to the list

   numbers.push_back(5);  // Add at the end

   numbers.push_back(10);

   numbers.push_front(1); // Add at the front


   cout << "List elements after adding values:\n";

   for (int num : numbers) {

      cout << num << " ";

   }

   cout << endl;
```

```
    return 0;
}
```

Output:

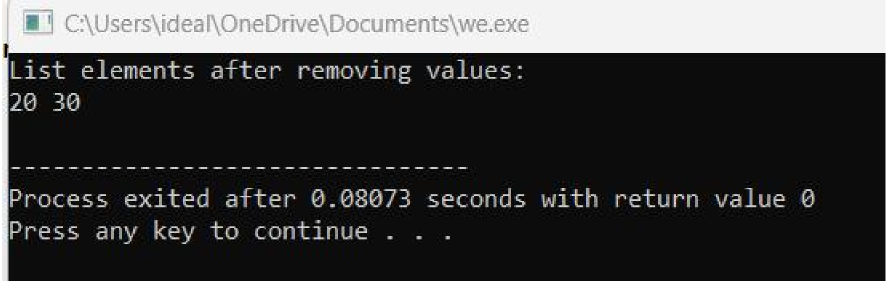


```
List elements after adding values:
1 5 10

--------------------------------
Process exited after 0.0799 seconds with return value 0
Press any key to continue . . .
```

Code 3:

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

int main() {
    list<int> numbers = {10, 20, 30, 40};

    // Removing the first and last elements
    numbers.pop_front();
    numbers.pop_back();

    cout << "List elements after removing values:\n";
    for (int num : numbers) {
        cout << num << " ";
    }
    cout << endl;


    return 0;
}
```

Output:

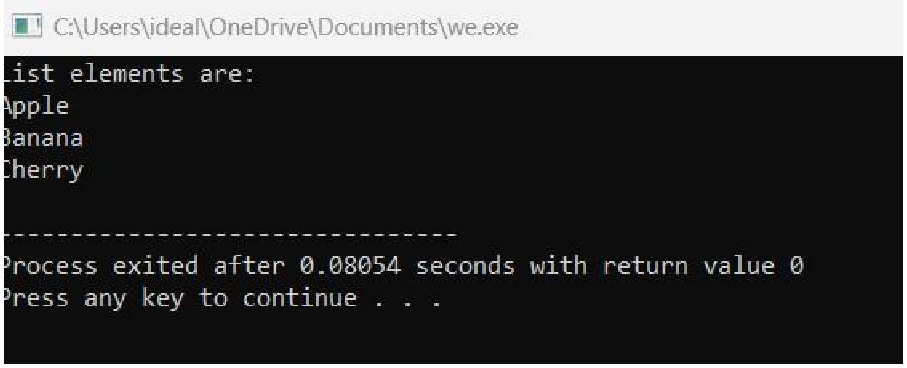


```
List elements after removing values:
20 30

--------------------------------
Process exited after 0.08073 seconds with return value 0
Press any key to continue . . .
```

Code 4:

```
#include <iostream>

#include <list>

using namespace std;


int main() {

    list<string> fruits = {"Apple", "Banana", "Cherry"};


    cout << "List elements are:\n";

    for (list<string>::iterator it = fruits.begin(); it != fruits.end(); ++it) {

        cout << *it << endl;

    }
    return 0;
}
```

Output:

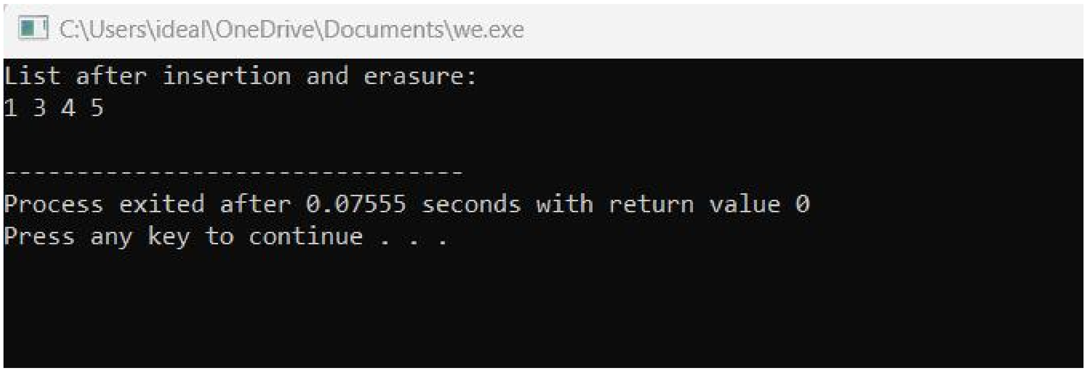


```
List elements are:
Apple
Banana
Cherry

--------------------------------
Process exited after 0.08054 seconds with return value 0
Press any key to continue . . .
```

Code 5:

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


int main() {
    list<int> numbers = {1, 2, 4, 5};


    // Insert 3 before the 3rd element
    auto it = numbers.begin();
    advance(it, 2);  // Move iterator to the 3rd position
    numbers.insert(it, 3);


    // Erase the 2nd element
    it = numbers.begin();
    advance(it, 1);  // Move iterator to the 2nd position
    numbers.erase(it);


    cout << "List after insertion and erasure:\n";
    for (int num : numbers) {
        cout << num << " ";
    }
    cout << endl;


    return 0;
}
```

Output:



```
List after insertion and erasure:
1 3 4 5

--------------------------------
Process exited after 0.07555 seconds with return value 0
Press any key to continue . . .
```

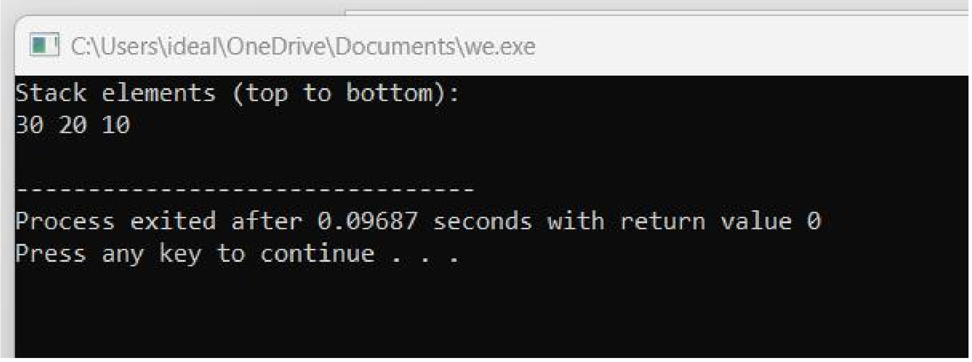
Lab 05

Stack

code 1:

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

int main() {
   stack<int> numbers;

   // Pushing elements onto the stack
   numbers.push(10);
   numbers.push(20);
   numbers.push(30);

   cout << "Stack elements (top to bottom):\n";
   while (!numbers.empty()) {
      cout << numbers.top() << " "; // Accessing the top element
      numbers.pop(); // Removing the top element
   }
   cout << endl;

   return 0;
}
```

Output :



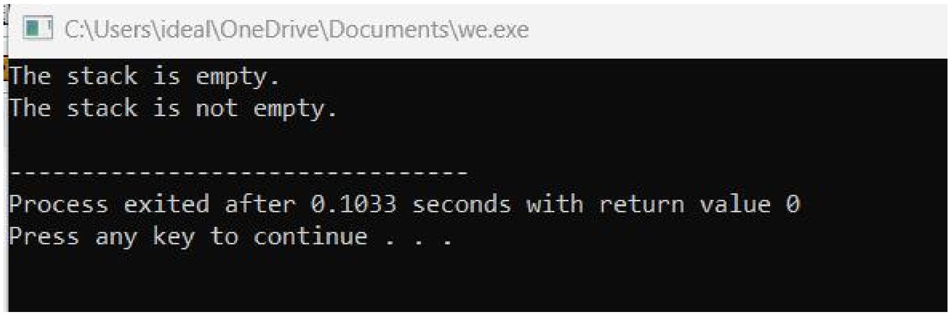
C:\Users\ideal\OneDrive\Documents\we.exe

```
Stack elements (top to bottom):
30 20 10

--------------------------------
Process exited after 0.09687 seconds with return value 0
Press any key to continue . . .
```

Code 2:

```
#include <iostream>

#include <stack>

using namespace std;


int main() {

    stack<int> numbers;


    // Checking if the stack is empty

    if (numbers.empty()) {

        cout << "The stack is empty.\n";

    } else {

        cout << "The stack is not empty.\n";

    }


    // Push an element and check again

    numbers.push(100);
    if (numbers.empty()) {
        cout << "The stack is empty.\n";
    } else {
        cout << "The stack is not empty.\n";
    }

    return 0;
}
```

Output:

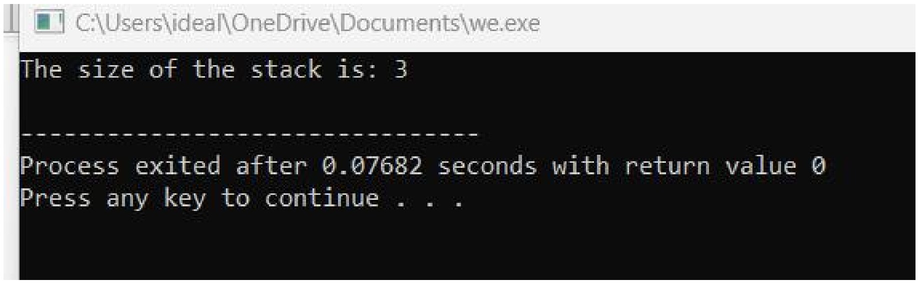


```
The stack is empty.
The stack is not empty.

--------------------------------
Process exited after 0.1033 seconds with return value 0
Press any key to continue . . .
```

Code 3:

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

int main() {
    stack<int> numbers;

    numbers.push(5);
    numbers.push(15);
    numbers.push(25);
    cout << "The size of the stack is: " << numbers.size() << endl;


    return 0;
}
```

Output:

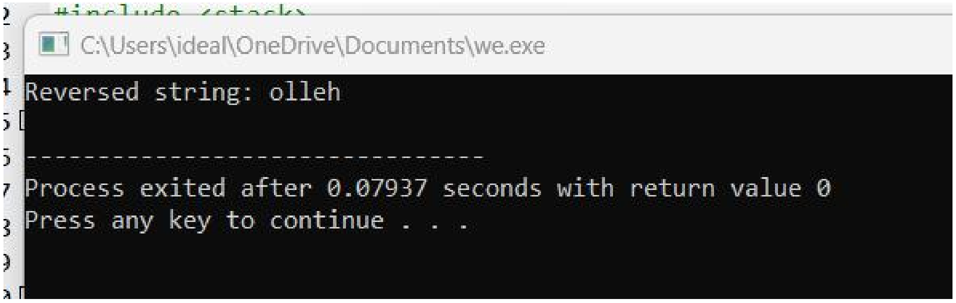


```
The size of the stack is: 3

--------------------------------
Process exited after 0.07682 seconds with return value 0
Press any key to continue . . .
```

Code 4:

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


int main() {
    string input = "hello";
    stack<char> charStack;


    // Push characters onto the stack
    for (char c : input) {
        charStack.push(c);
    }
    cout << "Reversed string: ";
    // Pop characters from the stack to reverse the string
    while (!charStack.empty()) {
        cout << charStack.top();
        charStack.pop();
    }
    cout << endl;

    return 0;

}
```

Output:

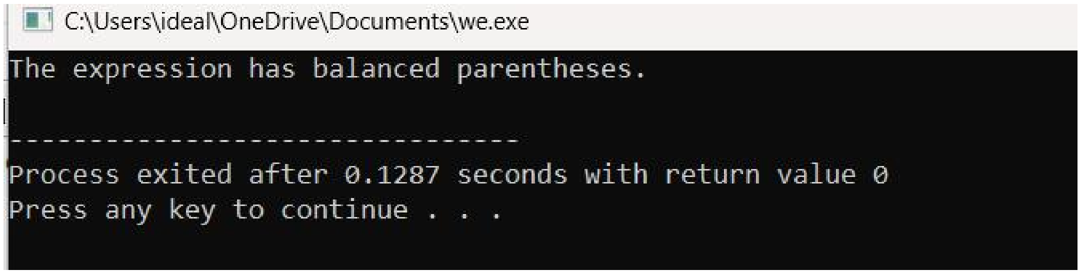


```
Reversed string: olleh

--------------------------------
Process exited after 0.07937 seconds with return value 0
Press any key to continue . . .
```

Code 5:

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

bool isBalanced(const string& expression) {
    stack<char> brackets;

    for (char c : expression) {
        if (c == '(') {
            brackets.push(c);
        } else if (c == ')') {
            if (brackets.empty()) {
                return false;
            }
            brackets.pop();
        }
    }

    return brackets.empty();
}

int main() {
    string expression = "(a + b) * (c - d)";
    if (isBalanced(expression)) {
        cout << "The expression has balanced parentheses.\n";
    } else {
        cout << "The expression does not have balanced parentheses.\n";
    }
    return 0;
}
```

Output :



```
C:\Users\ideal\OneDrive\Documents\we.exe
The expression has balanced parentheses.

--------------------------------
Process exited after 0.1287 seconds with return value 0
Press any key to continue . . .
```

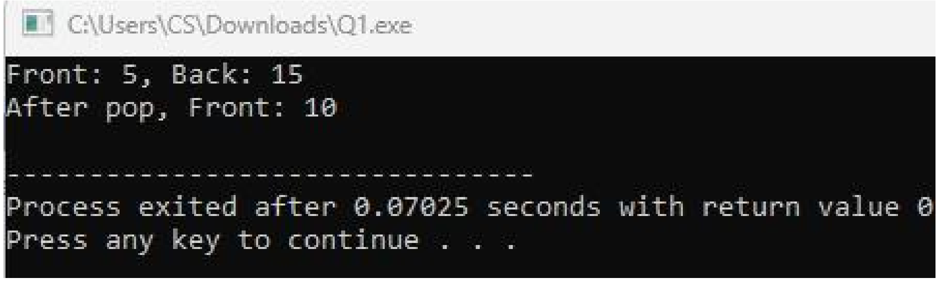
Lab 06

Code 1:

Pop front, back:

```
#include <iostream>
#include <queue>
using namespace std;
int main() {
    queue<int> q;
    q.push(5);
    q.push(10);
    q.push(15);
    cout << "Front: " << q.front() << ", Back: " << q.back() << endl;
    q.pop();
    cout << "After pop, Front: " << q.front() << endl;
    return 0;
}
```

Output:



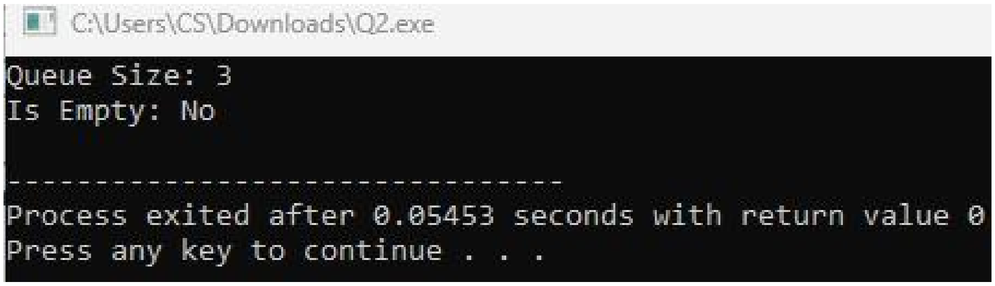
```
Front: 5, Back: 15
After pop, Front: 10

--------------------------------
Process exited after 0.07025 seconds with return value 0
Press any key to continue . . .
```

Code 2:

Queue size:

```
#include <iostream>

#include <queue>

using namespace std;


int main() {

    queue<int> q;

    q.push(1); q.push(2); q.push(3);

    cout << "Queue Size: " << q.size() << endl;

    cout << "Is Empty: " << (q.empty() ? "Yes" : "No") << endl;

    return 0;

}
```

Output:



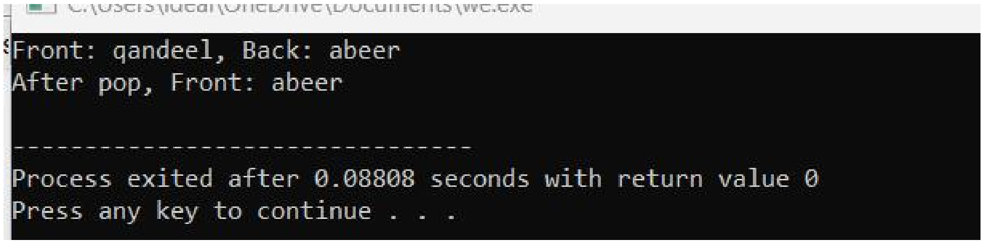
```
Queue Size: 3
Is Empty: No

--------------------------------
Process exited after 0.05453 seconds with return value 0
Press any key to continue . . .
```

Code 3:

Push elements:

```
#include <iostream>
#include <queue>
using namespace std;
int main() {
    queue<string> q;
    q.push("qandeel");
    q.push("abeer");
    cout << "Front: " << q.front() << ", Back: " << q.back() << endl;
    q.pop();
    cout << "After pop, Front: " << q.front() << endl;
    return 0;
}
```

Output:



```
Front: qandeel, Back: abeer
After pop, Front: abeer

--------------------------------
Process exited after 0.08808 seconds with return value 0
Press any key to continue . . .
```

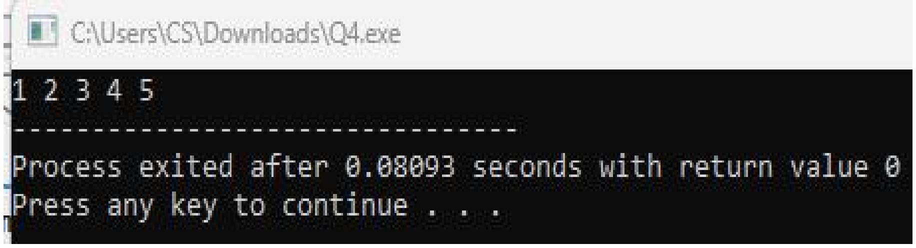
Code 4:

Empty queue:

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

```
int main() {
    queue<int> q;
    for (int i = 1; i <= 5; ++i) q.push(i);
    while (!q.empty()) {
        cout << q.front() << " ";
        q.pop();
    }
    return 0;
}
```

Output:



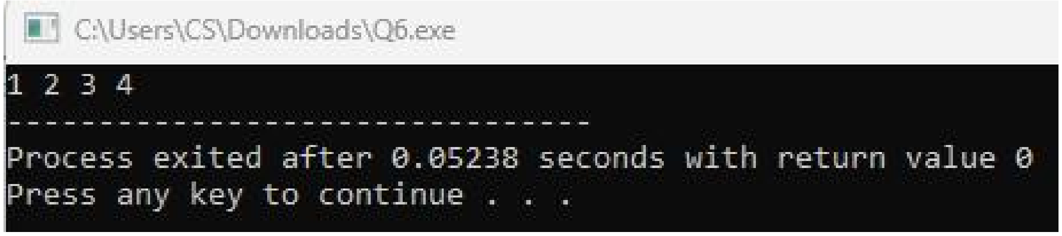
C:\Users\CS\Downloads\Q4.exe

```
1 2 3 4 5
--------------------------------
Process exited after 0.08093 seconds with return value 0
Press any key to continue . . .
```

Code 5:

```
#include <iostream>
#include <queue>
using namespace std;
int main() {
    queue<int> q1, q2;
    q1.push(1); q1.push(2);
    q2.push(3); q2.push(4);
    while (!q1.empty()) { cout << q1.front() << " "; q1.pop(); }
    while (!q2.empty()) { cout << q2.front() << " "; q2.pop(); }
    return 0;
}
```

Output:



C:\Users\CS\Downloads\Q6.exe

```
1 2 3 4
--------------------------------
Process exited after 0.05238 seconds with return value 0
Press any key to continue . . .
```

Lab 07

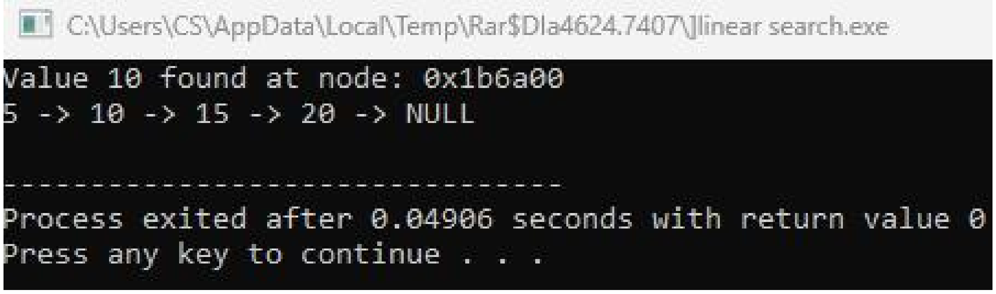
Single Link list

Code

Linear search:

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


class Node {
public:
    int val;
    Node* next;
    Node(int data) {
        val = data;
        next = NULL;
    }
};
void insert(Node*& head, int data) {
    Node* newNode = new Node(data);
    if (head == NULL) {
        head = newNode;
        return;
    }
    Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}

void search(Node* head, int v) {
    Node* temp = head;
    bool found = false;
    while (temp != NULL) {
        if (temp->val == v) {
            cout << "Value " << v << " found at node: " << temp << endl;
            found = true;
        }
        temp = temp->next;
    }
    if (!found) {
        cout << "Value " << v << " not found in the list." << endl;
    }
}

void displayList(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        cout << temp->val << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}

int main() {
    Node* head = NULL;
    insert(head, 5);
    insert(head, 10);
    insert(head, 15);
    insert(head, 20);

    search(head, 10);
    displayList(head);

    return 0;
}
```

Output:


C:\Users\CS\AppData\Local\Temp\Rar$DIa4624.7407\Jlinear search.exe

```
Value 10 found at node: 0x1b6a00
5 -> 10 -> 15 -> 20 -> NULL

--------------------------------
Process exited after 0.04906 seconds with return value 0
Press any key to continue . . .
```

<u>Deletion</u>

Code 1:

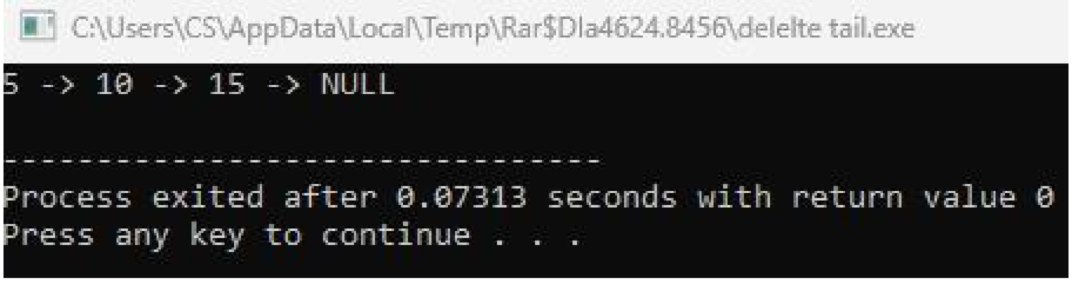
Delete tail:

```
#include <iostream>
```

```
using namespace std;

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

void insert(Node*& head, int data) {
    Node* newNode = new Node(data);
    if (head == NULL) {
        head = newNode;
        return;
    }
    Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}

void delTail(Node* head){
            Node* secondlast=head;
            while (secondlast->next->next != NULL){
```

```
                        secondlast=secondlast->next;
            }
            Node* temp=secondlast->next;
            secondlast->next = NULL;
            delete temp;
}

void displayList(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        cout << temp->val << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}

int main() {
    Node* head = NULL;
    insert(head, 5);
    insert(head, 10);
    insert(head, 15);
    insert(head, 20);
    delTail(head);
    displayList(head);
}
```

Output:



C:\Users\CS\AppData\Local\Temp\Rar$DIa4624.8456\delelte tail.exe

```
5 -> 10 -> 15 -> NULL

--------------------------------
Process exited after 0.07313 seconds with return value 0
Press any key to continue . . .
```

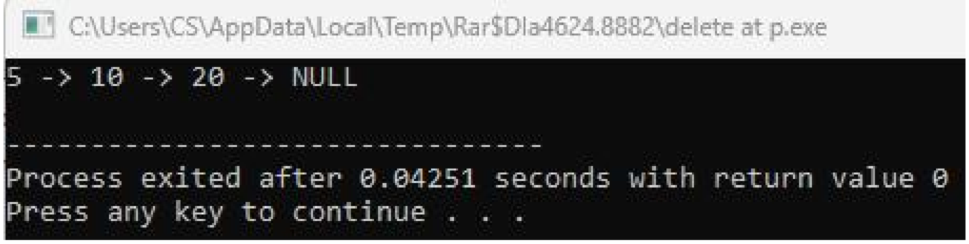
Code 2:

Delete at position:

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

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

void insert(Node*& head, int data) {
    Node* newNode = new Node(data);
    if (head == NULL) {
        head = newNode;
        return;
    }
    Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}

void delatP(Node* &head, int pos){
            Node* prev=head;
            int currentpos=0;
            while (currentpos != pos-1){
                        prev = prev->next;
                        currentpos++;
            }
            Node* temp = prev->next;
            prev->next= prev->next->next;
            delete temp;
}

void displayList(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        cout << temp->val << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}
```

```
int main() {
    Node* head = NULL;
    insert(head, 5);
    insert(head, 10);
    insert(head, 15);
    insert(head, 20);
    delatP(head,2);
    displayList(head);
}
```

Output:



C:\Users\CS\AppData\Local\Temp\Rar$DIa4624.8882\delete at p.exe

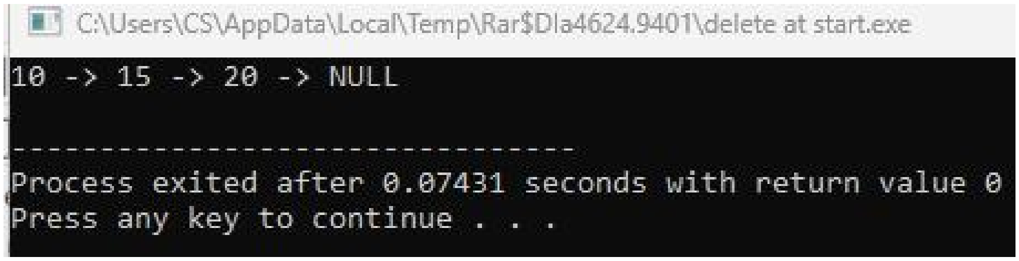
```
5 -> 10 -> 20 -> NULL

--------------------------------
Process exited after 0.04251 seconds with return value 0
Press any key to continue . . .
```

Code 3:

Delete at start:

```
#include <iostream>
using namespace std;
class Node {
public:
    int val;
    Node* next;
    Node(int data) {
        val = data;
        next = NULL;
    }
};

void insert(Node*& head, int data) {
    Node* newNode = new Node(data);
    if (head == NULL) {
        head = newNode;
        return;
    }
    Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}

void delatstart(Node*& head) {
    if (head == NULL) return;
    Node* temp = head;
    head = head->next;
    delete temp;
}

void displayList(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        cout << temp->val << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}

int main() {
    Node* head = NULL;
    insert(head, 5);
    insert(head, 10);
    insert(head, 15);
    insert(head, 20);
    delatstart(head);
    displayList(head);
}
```

Output:


C:\Users\CS\AppData\Local\Temp\Rar$DIa4624.9401\delete at start.exe

```
10 -> 15 -> 20 -> NULL

--------------------------------
Process exited after 0.07431 seconds with return value 0
Press any key to continue . . .
```

Code 4:

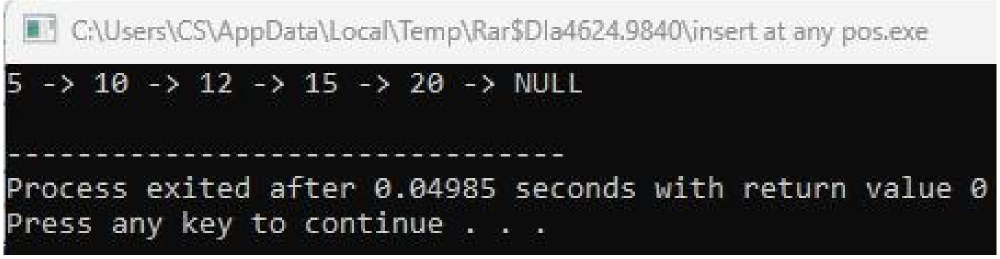
Insert at position:

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

class Node {
```

```
public:
    int val;
    Node* next;
    Node(int data) {
        val = data;
        next = NULL;
    }
};

void insert(Node*& head, int data) {
    Node* newNode = new Node(data);
    if (head == NULL) {
        head = newNode;
        return;
    }
    Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}
void insertatstart(Node* &head, int data){
            Node* newnode = new Node(data);
            newnode->next = head;
            head = newnode;
}
void insertatP(Node* &head, int val, int pos){
            if(pos==0){
```

```
                        insertatstart(head,val);
                        return;
            }
            Node* newnode = new Node(val);
            Node* temp = head;
            int currentpos=0;
            while (currentpos!=pos-1){
                        temp=temp->next;
                        currentpos++;
            }
            newnode->next=temp->next;
            temp->next = newnode;
}
void displayList(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        cout << temp->val << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}

int main() {
    Node* head = NULL;
    insert(head, 5);
    insert(head, 10);
    insert(head, 15);
    insert(head, 20);
    insertatP(head,12,2);
    displayList(head);


    return 0;
}
```

Output:


C:\Users\CS\AppData\Local\Temp\Rar$DIa4624.9840\insert at any pos.exe

```
5 -> 10 -> 12 -> 15 -> 20 -> NULL

--------------------------------
Process exited after 0.04985 seconds with return value 0
Press any key to continue . . .
```

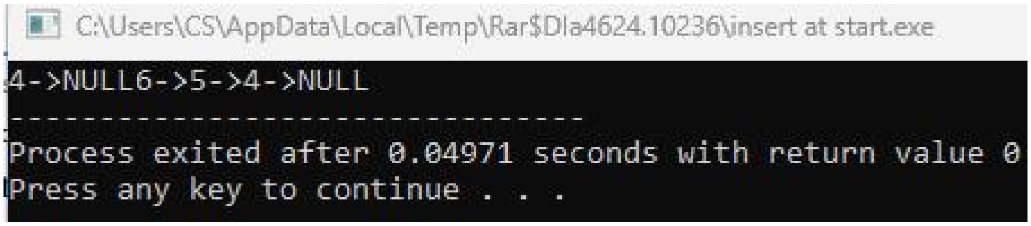
Insertion

Code 1:

Insert at start:

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


class Node {
public:
    int val;
    Node* next;
    Node(int data) {
        val = data;
        next = NULL;
    }
};
void insertatstart(Node* &head, int data){
    Node* newnode = new Node(data);
    newnode->next = head;
    head = newnode;
}
void display(Node* head){
    Node* temp = head;
    while (temp!=NULL){
        cout << temp->val << "->";
        temp=temp->next;
    }
    cout << "NULL";
}
int main (){
    Node* head = NULL;
    insertatstart(head,4);
    display(head);
    insertatstart(head,5);
    insertatstart(head,6);
    display(head);
}
```

Output:


C:\Users\CS\AppData\Local\Temp\Rar$DIa4624.10236\insert at start.exe

```
4->NULL6->5->4->NULL
--------------------------------
Process exited after 0.04971 seconds with return value 0
Press any key to continue . . .
```

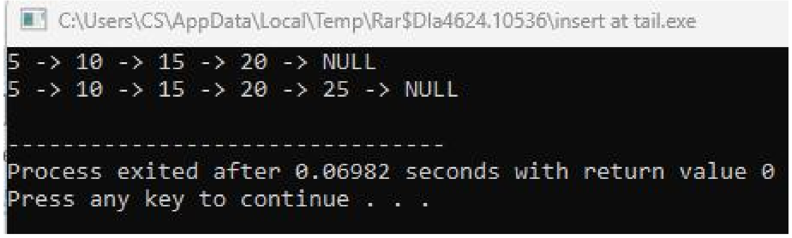
Code 2:

Insert at tail:

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

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

void insert(Node*& head, int data) {
    Node* newNode = new Node(data);
    if (head == NULL) {
        head = newNode;
        return;
    }
    Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}
void insertAtTail(Node* &head,int val){
```

```
            Node* newnode=new Node(val);
            Node* temp=head;
            while(temp->next!=NULL){
                        temp = temp->next;
            }
            temp->next=newnode;
}
void displayList(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        cout << temp->val << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}
int main() {
    Node* head = NULL;
    insert(head, 5);
    insert(head, 10);
    insert(head, 15);
    insert(head, 20);

    displayList(head);
    insertAtTail(head,25);
    displayList(head);
    return 0;
}
```

Output:



C:\Users\CS\AppData\Local\Temp\Rar$DIa4624.10536\insert at tail.exe

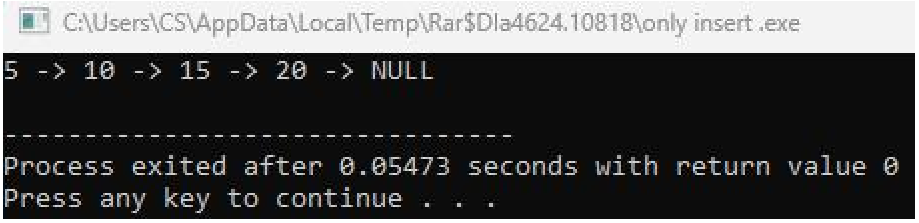
```
5 -> 10 -> 15 -> 20 -> NULL
5 -> 10 -> 15 -> 20 -> 25 -> NULL

--------------------------------
Process exited after 0.06982 seconds with return value 0
Press any key to continue . . .
```

Code 3:

Only insert:

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

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

void insert(Node*& head, int data) {
    Node* newNode = new Node(data);
    if (head == NULL) {
        head = newNode;
        return;
    }
    Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}
void displayList(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        cout << temp->val << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}

int main() {
    Node* head = NULL;
    insert(head, 5);
    insert(head, 10);
    insert(head, 15);
    insert(head, 20);

    displayList(head);
    return 0;
}
```

Output:



C:\Users\CS\AppData\Local\Temp\Rar$DIa4624.10818\only insert .exe

```
5 -> 10 -> 15 -> 20 -> NULL

--------------------------------
Process exited after 0.05473 seconds with return value 0
Press any key to continue . . .
```

Lab 8

Circular link list

Insertion

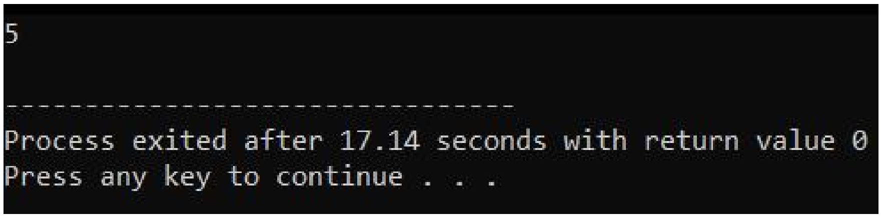
Code 1:

Inert at beginning:

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

struct Node {
   int data;
   Node* next;
};

void insertBegin(Node** head, int data) {
   Node* newNode = new Node();
   newNode->data = data;
   newNode->next = NULL;

   if (*head == NULL) {
      newNode->next = newNode;
      *head = newNode;
   } else {
      Node* temp = *head;
      while (temp->next != *head) {
         temp = temp->next;
      }
      temp->next = newNode;
```

```
        newNode->next = *head;
        *head = newNode;
    }
}

void display(Node* head) {
    if (head == NULL) {
        cout << "List is empty." << endl;
        return;
    }
    Node* temp = head;
    do {
        cout << temp->data << " ";
        temp = temp->next;
    } while (temp != head);
    cout << endl;
}
int main() {
    Node* head = NULL;
    insertBegin(&head, 45);
    insertBegin(&head, 80);
    insertBegin(&head, 15);
    display(head);
    return 0;
}
```

Output:



```
5

--------------------------------
Process exited after 17.14 seconds with return value 0
Press any key to continue . . .
```

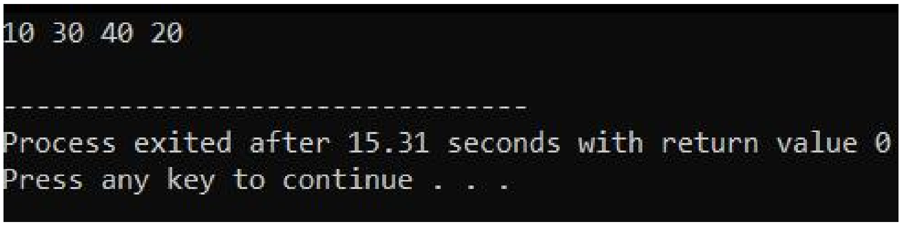
Code 2:

Insert at beginning:

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

struct Node {
    int data;
    Node* next;
};

void insertMid(Node** head, int data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = NULL;

    if (*head == NULL) {
        newNode->next = newNode;
        *head = newNode;
    } else {
        Node* slow = *head;
        Node* fast = *head;
        while (fast->next != *head && fast->next->next != *head) {
            slow = slow->next;
            fast = fast->next->next;
        }
        newNode->next = slow->next;
        slow->next = newNode;
    }
}

void display(Node* head) {
    if (head == NULL) {
        cout << "List is empty." << endl;
        return;
    }
    Node* temp = head;
    do {
        cout << temp->data << " ";
        temp = temp->next;
    } while (temp != head);
    cout << endl;}
int main() {
    Node* head = NULL;
    insertMid(&head, 10);
    insertMid(&head, 20);
    insertMid(&head, 30);
    insertMid(&head, 40);
    display(head);
    return 0;
```

```
}
```

Output:



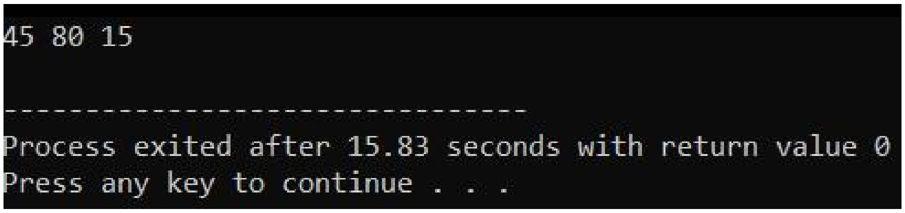
```
10 30 40 20

--------------------------------
Process exited after 15.31 seconds with return value 0
Press any key to continue . . .
```

Code 3:

Insert at end:

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

struct Node {
    int data;
    Node* next;
};

void insertEnd(Node** head, int data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = NULL;

    if (*head == NULL) {
        newNode->next = newNode;
        *head = newNode;
    } else {
        Node* temp = *head;
        while (temp->next != *head) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->next = *head;
    }
}
void display(Node* head) {
    if (head == NULL) {
        cout << "List is empty." << endl;
        return;
    }
    Node* temp = head;
    do {
        cout << temp->data << " ";
        temp = temp->next;
    } while (temp != head);
    cout << endl;
}
int main() {
    Node* head = NULL;
    insertEnd(&head, 45);
    insertEnd(&head, 80);
    insertEnd(&head, 15);
    display(head);
    return 0;
}
```

Output:



```
45 80 15

--------------------------------
Process exited after 15.83 seconds with return value 0
Press any key to continue . . .
```

Deletion

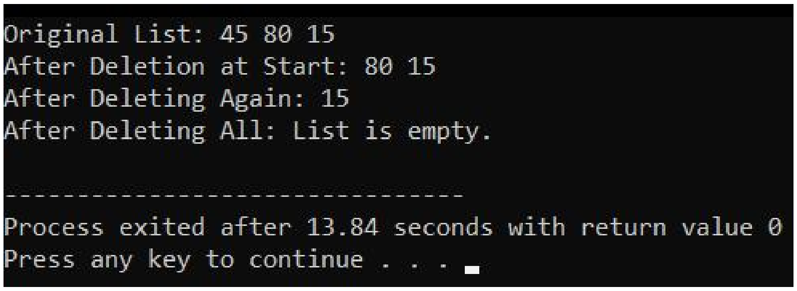
Code 1:

Delete at start:

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

struct Node {
    int data;
    Node* next;
};

void deleteStart(Node** head) {
    if (*head == NULL) {
        cout << "List is empty." << endl;
        return;
    }

    Node* temp = *head;

    if ((*head)->next == *head) {
        delete *head;
        *head = NULL;
        return;
    }

    Node* last = *head;
    while (last->next != *head) {
        last = last->next;
    }

    Node* newHead = (*head)->next;
    last->next = newHead;

    delete *head;
    *head = newHead;
}

void insertEnd(Node** head, int data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = NULL;

    if (*head == NULL) {
        newNode->next = newNode;
        *head = newNode;
    } else {
        Node* temp = *head;
        while (temp->next != *head) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->next = *head;
    }
}
```

```
void display(Node* head) {
    if (head == NULL) {
        cout << "List is empty." << endl;
        return;
    }
    Node* temp = head;
    do {
        cout << temp->data << " ";
        temp = temp->next;
    } while (temp != head);
    cout << endl;
}

int main() {
    Node* head = NULL;

    insertEnd(&head, 45);
    insertEnd(&head, 80);
    insertEnd(&head, 15);

    cout << "Original List: ";
    display(head);

    deleteStart(&head);
    cout << "After Deletion at Start: ";
    display(head);

    deleteStart(&head);
    cout << "After Deleting Again: ";
    display(head);

    deleteStart(&head);
    cout << "After Deleting All: ";
    display(head);

    return 0;
}
```

Output:



```
Original List: 45 80 15
After Deletion at Start: 80 15
After Deleting Again: 15
After Deleting All: List is empty.

--------------------------------
Process exited after 13.84 seconds with return value 0
Press any key to continue . . .
```

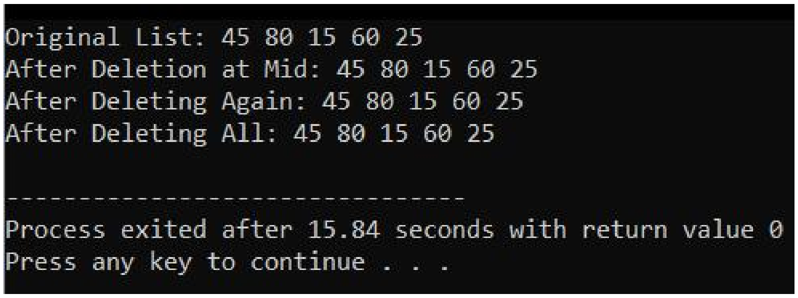
Code 2:

Delete at mid:

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

struct Node {
    int data;
    Node* next;
};

void deleteMid(Node** head) {
    if (*head == NULL) {
        cout << "List is empty." << endl;
        return;
    }

    if ((*head)->next == *head) {
        delete *head;
        *head = NULL;
        return;
    }

    Node* slow = *head;
    Node* fast = *head;
    Node* prev = NULL;

    while (fast != *head && fast->next != *head) {
        prev = slow;
        slow = slow->next;
        fast = fast->next->next;
    }

    if (prev != NULL) {
        prev->next = slow->next;
        if (slow == *head) {
            *head = slow->next;
        }
        delete slow;
    }
}
```

```
void insertEnd(Node** head, int data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = NULL;

    if (*head == NULL) {
        newNode->next = newNode;
        *head = newNode;
    } else {
        Node* temp = *head;
        while (temp->next != *head) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->next = *head;
    }
}

void display(Node* head) {
    if (head == NULL) {
        cout << "List is empty." << endl;
        return;
    }
    Node* temp = head;
    do {
        cout << temp->data << " ";
        temp = temp->next;
    } while (temp != head);
    cout << endl;
}

int main() {
    Node* head = NULL;

    insertEnd(&head, 45);
    insertEnd(&head, 80);
    insertEnd(&head, 15);
    insertEnd(&head, 60);
    insertEnd(&head, 25);

    cout << "Original List: ";
    display(head);

    deleteMid(&head);
    cout << "After Deletion at Mid: ";
    display(head);

    deleteMid(&head);
    cout << "After Deleting Again: ";
    display(head);

    deleteMid(&head);
    cout << "After Deleting All: ";
    display(head);
    return 0;
}
```

Output:



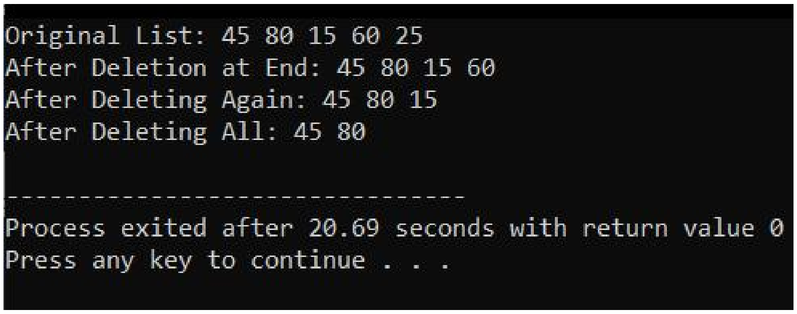
```
Original List: 45 80 15 60 25
After Deletion at Mid: 45 80 15 60 25
After Deleting Again: 45 80 15 60 25
After Deleting All: 45 80 15 60 25

--------------------------------
Process exited after 15.84 seconds with return value 0
Press any key to continue . . .
```

Code 3:

Delete at end:

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

struct Node {
    int data;
    Node* next;
};

void deleteEnd(Node** head) {
    if (*head == NULL) {
        cout << "List is empty." << endl;
        return;
    }

    if ((*head)->next == *head) {
        delete *head;
        *head = NULL;
        return;
    }

    Node* temp = *head;
    Node* prev = NULL;

    while (temp->next != *head) {
        prev = temp;
        temp = temp->next;
    }

    prev->next = *head;
    delete temp;
}

void insertEnd(Node** head, int data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = NULL;

    if (*head == NULL) {
        newNode->next = newNode;
        *head = newNode;
    } else {
        Node* temp = *head;
        while (temp->next != *head) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->next = *head;
    }
}

void display(Node* head) {
    if (head == NULL) {
        cout << "List is empty." << endl;
        return;
    }
    Node* temp = head;
    do {
        cout << temp->data << " ";
        temp = temp->next;
    } while (temp != head);
    cout << endl;
}

int main() {
    Node* head = NULL;

    insertEnd(&head, 45);
    insertEnd(&head, 80);
    insertEnd(&head, 15);
    insertEnd(&head, 60);
    insertEnd(&head, 25);

    cout << "Original List: ";
    display(head);

    deleteEnd(&head);
    cout << "After Deletion at End: ";
    display(head);

    deleteEnd(&head);
    cout << "After Deleting Again: ";
    display(head);

    deleteEnd(&head);
    cout << "After Deleting All: ";
    display(head);

    return 0;
}
```

Output:



```
Original List: 45 80 15 60 25
After Deletion at End: 45 80 15 60
After Deleting Again: 45 80 15
After Deleting All: 45 80

--------------------------------
Process exited after 20.69 seconds with return value 0
Press any key to continue . . .
```

Lab 9

Doubly link list

Insertion

Code 1:

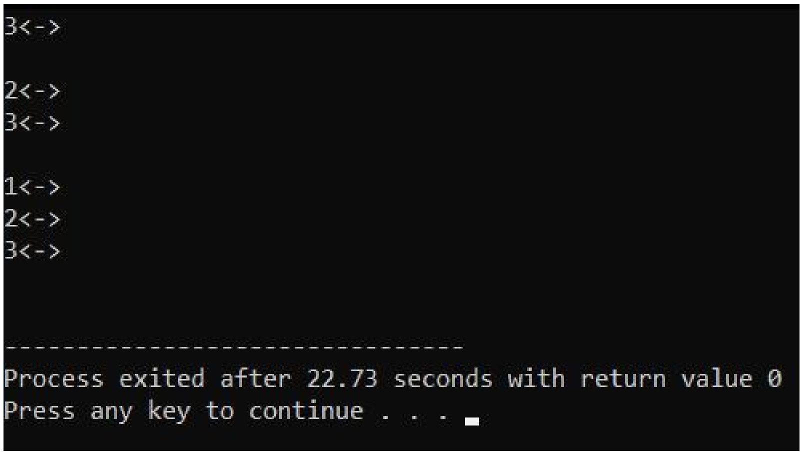
Insert at start:

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

class Node {
public:

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

class DOUBLELINKLIST{
public:
Node* head;
Node* tail;
DOUBLELINKLIST(){
```

```
head = NULL;
tail = NULL;
}
    void insertHead(int val){
    Node* new_node = new Node(val);
if (head==NULL){
head=new_node;
tail= new_node;
return;
}

new_node-> next = head;
    head->prev=new_node;
    head=new_node;
    return;
}

void display(){
Node* temp=head;
while (temp!=NULL){
cout << temp->val << "<->"<<endl;
temp=temp->next;
}
cout << endl;
}
};

int main ()
```

```
{
DOUBLELINKLIST dll;
dll.insertHead(3);
dll.display();
dll.insertHead(2);
dll.display();
dll.insertHead(1);
dll.display();
}
```

Output:



```
3<->

2<->
3<->

1<->
2<->
3<->



--------------------------------
Process exited after 22.73 seconds with return value 0
Press any key to continue . . .
```

Code 2:

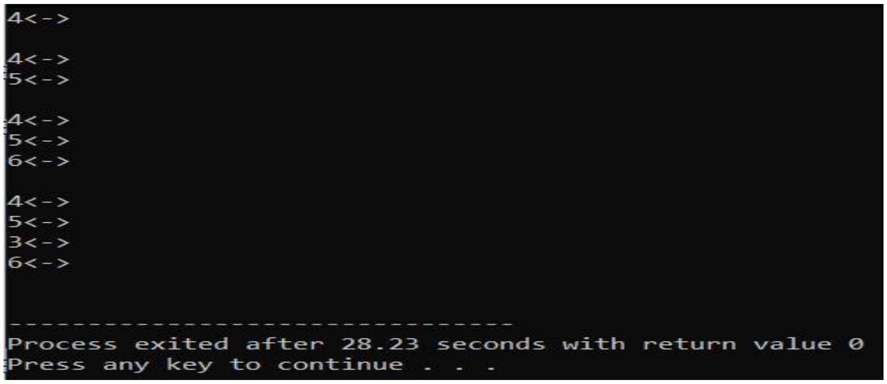
Insert at any position:

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

```
class Node {
public:

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

class DOUBLELINKLIST{
public:

Node* head;
Node* tail;
DOUBLELINKLIST(){
head = NULL;
tail = NULL;
}

   void insertend(int val){
   Node* new_node = new Node(val);
if (tail==NULL){
head=new_node;
tail= new_node;
return;
}
new_node-> prev = tail;
    tail->next=new_node;
    tail=new_node;
    return;
}
void display(){
Node* temp=head;
while (temp!=NULL){
cout << temp->val << "<->"<<endl;
temp=temp->next;
}
cout << endl;
}
    void insertatP(int val, int k){
    int count=0;
    Node* temp = head;
    while (count <(k-1)){
   temp = temp->next;
    count++;
}
Node* new_node = new Node(val);
new_node->next = temp->next;
temp->next=new_node;
new_node->prev=temp;
new_node->next->prev=new_node;
```

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

DOUBLELINKLIST dll;
dll.insertend(4);
dll.display();
dll.insertend(5);
dll.display();
dll.insertend(6);
dll.display();
dll.insertatP(3,2);
dll.display();
return 0;
}
```

Output:



```
4<->

4<->
5<->

4<->
5<->
6<->

4<->
5<->
3<->
6<->


--------------------------------
Process exited after 28.23 seconds with return value 0
Press any key to continue . . .
```

Deletion

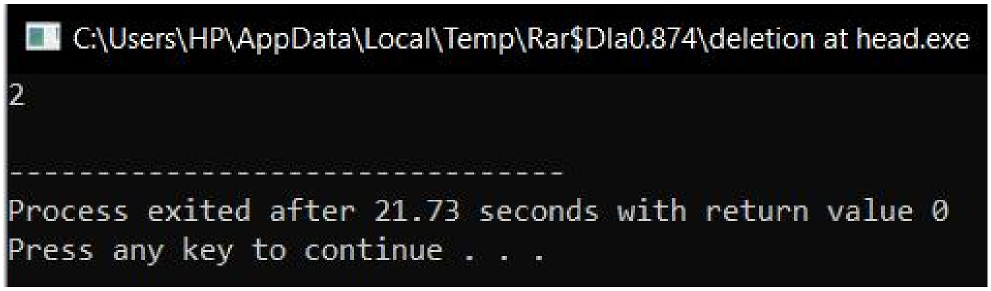
Code 1:

Delete at start:

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

class Node {
public:
    int val;
    Node* next;
    Node* prev;

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

class DOUBLELINKLIST {
public:
    Node* head;
    Node* tail;

    DOUBLELINKLIST() {
        head = NULL;
        tail = NULL;
    }

    void insert(int val) {
        Node* new_node = new Node(val);
        if (head == NULL) {
            head = new_node;
            tail = new_node;
        } else {
            tail->next = new_node;
            new_node->prev = tail;
            tail = new_node;
        }
    }

    void deleteAThead() {
        if (head == NULL) {
            return;
        }
        Node* temp = head;
        head = head->next;
        if (head == NULL) {
            tail = NULL;
        } else {
            head->prev = NULL;
        }
        delete temp;
    }
```

```
    void display() {
        Node* temp = head;
        while (temp != NULL) {
            cout << temp->val;
            if (temp->next != NULL) {
                cout << " <-> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
};

int main() {
    DOUBLELINKLIST dll;
    dll.insert(3);
    dll.insert(2);
    dll.deleteAThead();
    dll.display();
    return 0;
}
```

Output:



```
C:\Users\HP\AppData\Local\Temp\Rar$DIa0.874\deletion at head.exe
2

--------------------------------
Process exited after 21.73 seconds with return value 0
Press any key to continue . . .
```

Code:

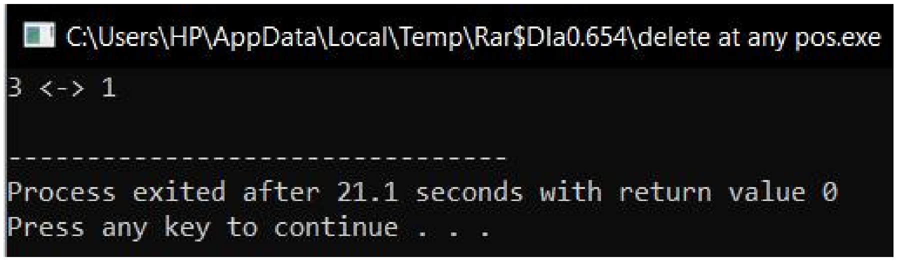
Delete at any position:

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

class Node {
public:
    int val;
    Node* next;
    Node* prev;

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

class DOUBLELINKLIST {
public:
    Node* head;
    Node* tail;

    DOUBLELINKLIST() {
        head = NULL;
        tail = NULL;
    }
```

```
void insert(int val) {
    Node* new_node = new Node(val);
    if (head == NULL) {
        head = new_node;
        tail = new_node;
    } else {
        tail->next = new_node;
        new_node->prev = tail;
        tail = new_node;
    }
}

void del(int p) {
    if (head == NULL) {
        cout << "List is empty." << endl;
        return;
    }

    Node* temp = head;
    int count = 1;

    while (temp != NULL && count < p) {
        temp = temp->next;
        count++;
    }

    if (temp == NULL) {
        cout << "Position out of bounds." << endl;
        return;
    }

    if (temp->prev != NULL)
        temp->prev->next = temp->next;

    if (temp->next != NULL)
        temp->next->prev = temp->prev;

    if (temp == head)
        head = temp->next;

    if (temp == tail)
        tail = temp->prev;

    delete temp;
}

void display() {
    Node* temp = head;
    while (temp != NULL) {
        cout << temp->val;
        if (temp->next != NULL) {
            cout << " <-> ";
        }
        temp = temp->next;
    }
```

```
        cout << endl;
    }
};

int main() {
    DOUBLELINKLIST dll;
    dll.insert(3);
    dll.insert(2);
    dll.insert(1);
    dll.del(2);
    dll.display();
    return 0;
}
```

Output:


C:\Users\HP\AppData\Local\Temp\Rar$DIa0.654\delete at any pos.exe

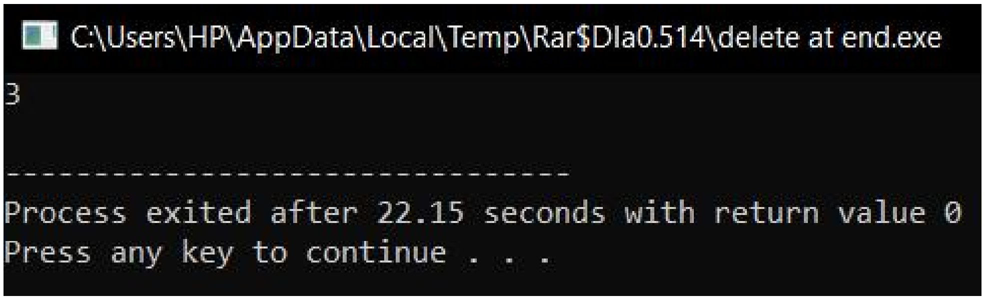
```
3 <-> 1

--------------------------------
Process exited after 21.1 seconds with return value 0
Press any key to continue . . .
```

Code:

Delete at end:

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

class Node {
public:
    int val;
    Node* next;
    Node* prev;

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

class DOUBLELINKLIST {
public:
    Node* head;
    Node* tail;

    DOUBLELINKLIST() {
        head = NULL;
        tail = NULL;
    }

    void insert(int val) {
    Node* new_node = new Node(val);
    if (head == NULL) {
        head = new_node;
        tail = new_node;
    } else {
        tail->next = new_node;
        new_node->prev = tail;
        tail = new_node;
    }
}



    void del(){
    if (head==NULL){
    return;
}
Node* temp = tail;
tail = tail->prev;
if(head==NULL){
tail=NULL;
else{
tail->next = NULL;
}
delete temp;
}

    void display() {
        Node* temp = head;
        while (temp != NULL) {
            cout << temp->val;
            if (temp->next != NULL) {
                cout << " <-> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
};

int main() {
    DOUBLELINKLIST dll;
    dll.insert(3);
    dll.insert(2);
    dll.del();
    dll.display();
    return 0;
}
```

Output:



C:\Users\HP\AppData\Local\Temp\Rar$DIa0.514\delete at end.exe

```
3

--------------------------------
Process exited after 22.15 seconds with return value 0
Press any key to continue . . .
```

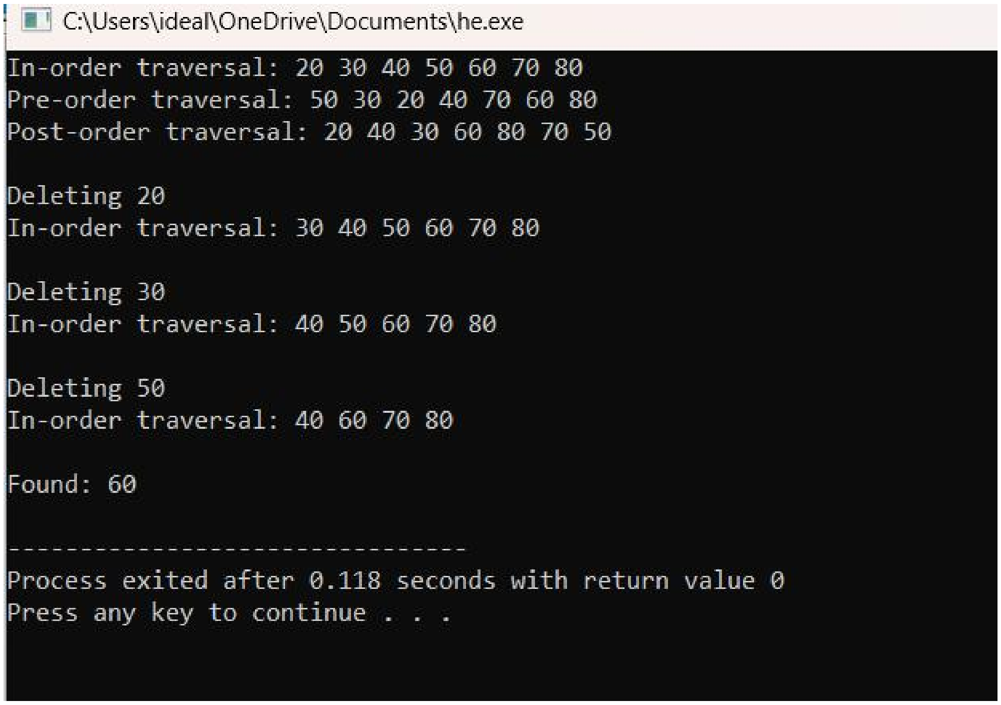
Binary search tree:

Code:

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

// Node structure
struct Node {
    int data;
    Node* left;
    Node* right;

    Node(int value) {
        data = value;
        left = nullptr;
        right = nullptr;
    }
};

// Insert function
Node* insert(Node* root, int key) {
    if (root == nullptr) {
        return new Node(key);
    }
    if (key < root->data) {
        root->left = insert(root->left, key);
    } else if (key > root->data) {
        root->right = insert(root->right, key);
    }

    return root;
}

// Search function
Node* search(Node* root, int key) {
    if (root == nullptr || root->data == key) {
        return root;
    }

    if (key < root->data) {
        return search(root->left, key);
    } else {
        return search(root->right, key);
    }
}

// Find the minimum value node
Node* findMin(Node* root) {
    while (root && root->left != nullptr) {
        root = root->left;
    }
    return root;
}

// Delete function
Node* deleteNode(Node* root, int key) {
    if (root == nullptr) {
        return root;
    }

    if (key < root->data) {
        root->left = deleteNode(root->left, key);
    } else if (key > root->data) {
        root->right = deleteNode(root->right, key);
    } else {
        if (root->left == nullptr) {
            Node* temp = root->right;
            delete root;
            return temp;
        } else if (root->right == nullptr) {
            Node* temp = root->left;
            delete root;
            return temp;
        }

        Node* temp = findMin(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }

    return root;
}

// In-order traversal
void inOrder(Node* root) {
    if (root != nullptr) {
        inOrder(root->left);
        cout << root->data << " ";
        inOrder(root->right);
    }
}

// Pre-order traversal
void preOrder(Node* root) {
    if (root != nullptr) {
        cout << root->data << " ";
        preOrder(root->left);
        preOrder(root->right);
    }
}

// Post-order traversal
void postOrder(Node* root) {
    if (root != nullptr) {
        postOrder(root->left);
        postOrder(root->right);
        cout << root->data << " ";
    }
}

// Main function to test the BST
int main() {
    Node* root = nullptr;

    root = insert(root, 50);
    root = insert(root, 30);
    root = insert(root, 20);
    root = insert(root, 40);
    root = insert(root, 70);
    root = insert(root, 60);
    root = insert(root, 80);

    cout << "In-order traversal: ";
    inOrder(root);
    cout << endl;

    cout << "Pre-order traversal: ";
    preOrder(root);
    cout << endl;

    cout << "Post-order traversal: ";
    postOrder(root);
    cout << endl;

    cout << "\nDeleting 20\n";
    root = deleteNode(root, 20);
    cout << "In-order traversal: ";
    inOrder(root);
    cout << endl;

    cout << "\nDeleting 30\n";
    root = deleteNode(root, 30);
    cout << "In-order traversal: ";
    inOrder(root);
```

```
    cout << endl;

    cout << "\nDeleting 50\n";
    root = deleteNode(root, 50);
    cout << "In-order traversal: ";
    inOrder(root);
    cout << endl;

    Node* found = search(root, 60);
    if (found != nullptr) {
        cout << "\nFound: " << found->data << endl;
    } else {
        cout << "\nNot Found" << endl;
    }

    return 0;
}
```

Output:



```
In-order traversal: 20 30 40 50 60 70 80
Pre-order traversal: 50 30 20 40 70 60 80
Post-order traversal: 20 40 30 60 80 70 50

Deleting 20
In-order traversal: 30 40 50 60 70 80

Deleting 30
In-order traversal: 40 50 60 70 80

Deleting 50
In-order traversal: 40 60 70 80

Found: 60

--------------------------------
Process exited after 0.118 seconds with return value 0
Press any key to continue . . .
```