STACK

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

class Stack
{
   vector<int> v;

public:
   void push(int val)
   {
      v.push_back(val);
   }

   void pop()
   {
      if (!v.empty())
      {
         v.pop_back();
      }
   }

   int top()
   {
      if (!v.empty())
      {
         return v[v.size() - 1];
      }

      return -1;
   }

   bool empty()
   {
      return v.size() == 0;
   }
};
int main()
{
   Stack s;
   s.push(230);
   s.push(21);
   s.push(1);
   s.push(64);
   while (!s.empty())
   {
      cout << s.top() << " ";
      s.pop();
   }

   return 0;
}
```

MATRIX

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

int main()
{
   int n;
```

```
cout << "1.Addition \n2.Multipication\nEnter your choice: ";
cin >> n;
int r1, c1, r2, c2;
cout << "Enter your 1st matrix Row & Col: ";
cin >> r1 >> c1;
int a[r1][c1];
cout << "Enter 1st matrix value : \n";
for (int i = 0; i < r1 * c1; i++)
{
   cin >> a[i / c1][i % c1];
}
cout << "Enter your 2nd matrix  Row & Col: ";
cin >> r2 >> c2;

int b[r2][c2];
cout << "Enter 2nd matrix value: \n";
for (int i = 0; i < r2 * c2; i++)
{
   cin >> b[i / c2][i % c2];
}

if (n == 1)
{
   if (r1 != r2 || c1 != c2)
   {
      cout << "Aditional is not posible !\n";
      return 0;
   }

   cout << "Result is : \n";
   for (int i = 0; i < r1 * c1; i++)
   {
      cout << a[i / c1][i % c1] + b[i / c1][i % c1] << " ";
      if ((i + 1) % c1 == 0)
      {
         cout << "\n";
      }
   }
}
else if (n == 2)
{
   if (c1 != r2)
   {
      cout << "Multipicition is not posible \n";
      return 0;
   }

   int c[r1][c2];
   for (int i = 0; i < r1; i++)
   {
      for (int j = 0; j < c2; j++)
      {
         c[i][j] = 0;
         for (int k = 0; k < c1; k++)
         {
            c[i][j] += a[i][k] * b[k][j];
         }
      }
   }

   cout << "Result is : \n";
   for (int i = 0; i < r1 * c2; i++)
   {
      cout << c[i / c2][i % c2] << " ";
      if ((i + 1) % c2 == 0)
      {
            cout << "\n";
          }
        }
    }

    return 0;
}
```

INSERTION SHORT

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

int main()
{
    int n;
    cout << "Enter the arary size: ";
    cin >> n;

    vector<int> v(n);
    for (auto &x : v)
    {
        cin >> x;
    }

    for (int i = 1; i < n; i++)
    {
        int key = v[i];
        int j = i - 1;
        while (j >= 0 and v[j] > key)
        {
            v[j + 1] = v[j];
            j--;
        }
        v[j + 1] = key;
    }

    for (auto x : v)
    {
        cout << x << " ";
    }
    return 0;
}
```

BINARY SEARCH

```
#include <bits/stdc++.h>
#include <iostream>
using namespace std;
int binary(vector<int> &v, int l, int h, int k)
{
    if (l > h)
    {
        return -1;
    }
    int mid = (l + h) / 2;
    if (v[mid] == k)
    {
        return mid;
    }

    else if (v[mid] > k)
    {
        return binary(v, l, mid - 1, k);
    }
    else
    {
        return binary(v, mid + 1, h, k);
    }
}

int main()
{
    int n;
    cout << "Enter the array size: ";
    cin >> n;
    vector<int> v(n);
    for (auto &x : v)
    {
        cin >> x;
    }
    cout << "\nEnter the key: ";
    int x;
    cin >> x;
    int index = binary(v, 0, n - 1, x);
    if (index != -1)
    {
        cout << "Found on -> " << index;
    }
    else
    {
        cout << "Not founde !";
    }

    return 0;
}
```

LINKED LIST

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

class List
{
    Node *head;
    Node *tail;

public:
    List()
    {
        head = tail = NULL;
    }

    void push_front(int val)
    {
        Node *newNode = new Node(val);
        if (head == NULL)
        {
            head = tail = newNode;
            return;
        }

        else
        {
            newNode->next = head;
            head = newNode;
        }
    }

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

    int search(int key)
    {
        Node *temp = head;
        int index = 0;
        while (temp != NULL)
        {
            if (temp->data == key)
            {
                return index;
            }
            temp = temp->next;
            index++;
        }
        return -1;
    }

    void print()
    {
        Node *temp = head;
        while (temp != NULL)
        {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << "\n";
    }
};
int main()
{
    List ll;
    ll.push_front(2);
    ll.push_front(1);
    ll.push_back(1234);
    ll.push_front(3);
    ll.push_front(6);
    ll.push_front(4);
    ll.push_front(34);
    ll.push_front(67);
    ll.push_back(100);
    ll.push_back(500);
    cout << ll.search(1234) << "\n";
    ll.print();

    return 0;
}
```

QUE

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

class Que
{
    vector<int> v;

public:
    void push(int val)
    {
        v.push_back(val);
    }
    void pop()
    {
        if (!v.empty())
        {
            v.erase(v.begin());
        }
    }

    int front()
    {
        if (!v.empty())
        {
            return v[0];
        }
        return -1;
    }

    bool empty()
    {
        return v.empty();
    }
};

int main()
{

    Que q;
    q.push(1);
    q.push(2);
    q.push(3);
    q.push(4);
    q.push(5);
    q.push(6);
    q.pop();
    q.pop();
    cout << q.empty();
    while (!q.empty())
    {
        cout << q.front() << " ";
        q.pop();
    }

    return 0;
}
```