NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

WEEK 1

```
/*

Q1.1-Given an array of nonnegative integers, design a linear algorithm and implement it
usinga program to find whether given key element is present in the array or not. Also, find
total number of comparisons for each input case. (Time Complexity = O(n), where n is the
size ofinput)
*/

#include <iostream>
using namespace std;
int linearsearch(int a[], int n, int key)

{

        int i;

        for (i = 0; i < n; i++)

        {

                if (a[i] == key)

                {

                        return i;

                }

        }


        return -1;

}


int main()

{

        int m, j;

        cout << "enter number";
        cin >> m;
        for (j = 0; j < m; j++)
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
    {

        int n, key;

        cout << "enter size";
        cin >> n;
        cout << "enter key";
        cin >> key;
        int i, a[n];

        for (i = 0; i < n; i++)

        {

            cin >> a[i];

        }



        int x = linearsearch(a, n, key);
        if (x == -1)
        {

            cout << "Not present";

        }

        else

        {

            cout << "Present" << x + 1;

        }

    }



    return 0;

}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
enter number 2

enter size 8

enter key 45

10

12

13

45

56

67

78

23

Present
4
enter size 3

enter key 56

1

2

3

Not present
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*

 SECTION :ML
CLASS ROLL NO :53

Q1.2-Given an already sorted array of positive integers, design an algorithm and implement
itusing a program to find whether given key element is present in the array or not.
(Time Complexity = O(nlogn), where n is the size of input)

*/

#include <iostream>
using namespace std;
void binarysearch(int a[], int n, int key)

{

        int l = 0, h = n - 1, m, i, pos, f = 0, c = 0;
        while (l <= h)
        {

                m = (l + h) / 2;
                c++;
                if (a[m] == key)

                {

                        f = 1;

                        break;

                }

                else if (a[m] > key)

                        h = m - 1;
                else if (a[m] < key)
                        l = m + 1;

        }



        if (f == 1)

        {

                cout << "present " << c << endl;

        }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
        else
        {
                cout << "not present " << endl;
        }
}


int main()
{
        int m, j;
        cout << "enter number";
        cin >> m;
        for (j = 0; j < m; j++)
        {
                int n, key;
                cout << "enter size";
                cin >> n;
                cout << "enter key";
                cin >> key;
                int i, a[n];
                for (i = 0; i < n; i++)
                {
                        cin >> a[i];
                }


                binarysearch(a, n, key);
        }


        return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
enter number 2

enter size 5

enter key 12

11

12

13

14

15

Present
3
enter size 6

enter key 10

11

12

10

9

80

76

present
1
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*

Q1.3-Given an already sorted array of positive integers, design an algorithm and implement
it using a program to find whether a given key element is present in the sorted array or not.
Foran array arr[n], search at the indexes arr[0], arr[2], arr[4],,arr[2k ] and so on. Once the
interval (arr[2k ] < key < arr[ 2k+1] ) is found, perform a linear search operation from the
index 2k to find the element key. (Complexity < O(n), where n is the number of elements
need to be scanned for searching):
 */

#include <iostream>
using namespace std;
int binarySearch(int arr[], int low, int high, int key)

{

        while (low <= high)

        {

                int mid = low + (high - low) / 2;
                if (arr[mid] == key)
                        return  mid;
                else if (arr[mid] < key)
                        low = mid + 1;

                else

                        high = mid - 1;

        }



        return -1;

}


int exponentialSearch(int arr[], int n, int key)

{

        if (arr[0] == key)
```

```
                return 0;
        int i = 1;
        while (i < n && arr[i] <= key)
                i = i * 2;
        return binarySearch(arr, i / 2, min(i, n - 1), key);
}


int main()
{
        int n, key, i;
        cout << "Enter size of the array: ";
        cin >> n;
        int arr[n];
        cout << "Enter sorted elements of the array: ";
        for (i = 0; i < n; i++)
                cin >> arr[i];
        cout << "Enter key to search: ";
        cin >> key;
        int result = exponentialSearch(arr, n, key);
        if (result == -1)
                cout << "Element not present in the array.\n";
        else
                cout << "Element found at index: " << result << endl;
        return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter size of the array: 5

Enter sorted elements of the array: 12
23
36

39

41

Enter key to search: 41
Element found at index: 4
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

WEEK 2

/*

QUESTION 2.1:
Given a sorted array of positive integers containing few duplicate elements, design an algorithm and implement it using a program to find whether the given key element is present in the array or not. If present, then also find the number of copies of given key. (Time Complexity = O(log n))

*/

```
#include <iostream>
using namespace std;
int binarySearch(int arr[], int l, int r, int x)

{

        if (r < l)

                return -1;

        int mid = l + (r - l) / 2;
        if (arr[mid] == x)
                return mid;
        if (arr[mid] > x)
                return binarySearch(arr, l, mid - 1, x);
        return binarySearch(arr, mid + 1, r, x);
}

int countOccurrences(int arr[], int n, int x)

{

    int ind = binarySearch(arr, 0, n - 1, x);
    if (ind == -1)
        return 0;
    int count = 1;
    int left = ind - 1;

    while (left >= 0 && arr[left] == x)

    {
        count++;
        left--;
    }
    int right = ind + 1;
    while (right < n && arr[right] == x)
    {
        count++;
        right++;
    }
    return count;
}
int main()
{
        int t,n,i,j,x;
        cout<<"enter number of test cases";
        cin>>t;
        for(i=0;i<t;i++)
        {
            cout<<"enter size";
            cin>>n;
            int a[n];
            for(j=0;j<n;j++)
            {
                cin>>a[j];
            }
            cout<<"enter key";
            cin>>x;
            cout << countOccurrences(a, n, x);
        }
        return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter number of test cases: 2
Enter size of array: 5
Enter elements of array (sorted): 1 2 2 3 4
Enter key to search: 2
Occurrences of 2: 2
Enter size of array: 4
Enter elements of array (sorted): 1 1 1 1
Enter key to search: 1
Occurrences of 1: 4
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 2. 2:
Given a sorted array of positive integers, design an algorithm and implement it using a
program to find three indices i, j, k such that arr[i] + arr[j] = arr[k].
*/
#include<iostream>
using namespace std;
int find(int a[],int n)
{
    int i,j,k;
    for(i=0;i<n-2;i++)
    {
        for(j=i+1;j<n-1;j++)
        {
            for(k=j+1;k<n;k++)
            {
                if(a[i]+a[j]==a[k])
                {
                    cout<<i+1<<" "<<j+1<<" "<<k+1;
                    return 1;
                }
            }
        }
    }
    return -1;
}
int main()
{
    int n,t,i,j;
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
    cout<<"enter number of test cases";
    cin>>t;
    for(i=0;i<t;i++)
    {
        cout<<"enter size";
        cin>>n;
        int a[n];
        for(j=0;j<n;j++)
        {
            cin>>a[j];
        }
        int x=find(a,n);
        if(x==-1)
            cout<<"No sequence found";
    }
    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter number of test cases: 3
Enter size of array: 5
Enter elements of array: 1 2 3 4 7
No sequence found
Enter size of array: 6

Enter elements of array: 3 6 9 12 15 18

1 2 4

Enter size of array: 7

Enter elements of array: 2 4 6 8 10 12 14
No sequence found
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 2.3:
Given an array of nonnegative integers, design an algorithm and a program to count the
number of pairs of integers such that their difference is equal to a given key, K.

*/
#include<iostream>
using namespace std;
int count(int a[],int n,int key)

{

    int c=0,i,j;
    for(i=0;i<n;i++)
    {

        for(j=i+1;j<n;j++)

        {

            if(a[i]-a[j]==key||a[j]-a[i]==key)

            {

                c++;

            }

        }

    }

    return c;

}

int main()

{

    int t,n,i,j,key;

    cout<<"enter number of test cases";
    cin>>t;
    for(i=0;i<t;i++)

    {
        cout<<"enter size";
        cin>>n;
        int a[n];
        for(j=0;j<n;j++)
            cin>>a[j];
        cout<<"enter key";
        cin>>key;
        int x=count(a,n,key);
        cout<<x<<endl;
    }

    return 0;

}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter number of test cases: 2
Enter size of array: 5
Enter elements of array: 1 5 3 2 4

Enter key: 1

Number of pairs with absolute difference 1: 3
Enter size of array: 4
Enter elements of array: 10 20 30 25

Enter key: 5

Number of pairs with absolute difference 5: 1
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

WEEK 3

```
/*
QUESTION 3.1:
Given an unsorted array of integers, design an algorithm and a program to sort the array using
insertion sort. Your program should be able to find number of comparisons and shifts ( shifts-
total number of times the array elements are shifted from their place) required for sorting the
array

*/

#include  <iostream>
using namespace std;
void insertionSort()
{

    int size;
    int key;
    int shift = 0;
    int com = 0;
    cout<<"Enter size of array::"<<endl;
    cin>>size;
    int arr[size];

    cout<<"Enter element::"<<endl;
    for(int i=0;i<size;i++)
    {

        cin>>arr[i];

    }

    for (int i = 1; i < size; i++)

    {

        key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key)

        {
            arr[j + 1] = arr[j];
            j = j - 1;
            shift ++ ;
            com ++ ;


        }
        arr[j + 1] = key;
        shift ++ ;
    }
    cout<<"Elements after sorting::"<<endl;
    for(int i=0;i<size;i++)
    {
        cout<<arr[i]<<endl;
    }
    cout<<"Number of comparisons::"<<com<<endl;
    cout<<"Number of shifts::"<<shift<<endl;


}


int main()
{
    int testcases;
    cout<<"Number of test cases::"<<endl;
    cin>>testcases;
    for(int i=0;i<testcases;i++)
    {
        cout<<"Testcase "<<i+1<<endl;
        insertionSort();
    }


}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Number of test cases: 2
Test case 1:
Enter size of array: 8

Enter elements: -23 65 -31 76 46 89 45 32

Elements after sorting: -31 -23 32 45 46 65 76 89
Number of comparisons: 13
Number of shifts: 20
Test case 2:
Enter size of array: 10

Enter elements: 54 65 34 76 78 97 46 32 51 21

Elements after sorting: 21 32 34 46 51 54 65 76 78 97
Number of comparisons: 28
Number of shifts: 37
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 3.2:
Given an unsorted array of integers, design an algorithm and implement a program to sort this
array using selection sort. Your program should also find number of comparisons and number
of swaps required.

*/

#include  <iostream>
using namespace std;
void selectionSort()
{

    int size;

    int swaps = 0;
    int com = 0;
    int i, j, min;
    cout<<"Enter size of array::"<<endl;
    cin>>size;
    int arr[size];

    cout<<"Enter element::"<<endl;
    for(i=0;i<size;i++)
    {

        cin>>arr[i];

    }

        for (i = 0; i < size - 1; i++)

    {

                min = i;

                for (j = i + 1; j < size; j++)

        {

                        if (arr[j] < arr[min])

                                min = j;
            com++;
```

```
                    }

                    if (min != i)

                              swap(arr[min], arr[i]);

        swaps++;

           }


     cout<<"Elements after sorting::"<<endl;
     for(int i=0;i<size;i++)
     {

        cout<<arr[i]<<endl;

     }

     cout<<"Number of comparisons::"<<com<<endl;
     cout<<"Number of swaps::"<<swaps<<endl;


  }



  int main()

  {

     int testcases;

     cout<<"Number of test cases::"<<endl;
     cin>>testcases;
     for(int i=0;i<testcases;i++)

     {

        cout<<"Testcase "<<i+1<<endl;
        selectionSort();
     }



  }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Number of test cases: 2
Test case 1:
Enter size of array: 8

Enter elements: -23 65 -31 76 46 89 45 32

Elements after sorting: -31 -23 32 45 46 65 76 89
Number of comparisons: 28
Number of swaps: 7
Test case 2:
Enter size of array: 10

Enter elements: 54 65 34 76 78 97 46 32 51 21

Elements after sorting: 21 32 34 46 51 54 65 76 78 97
Number of comparisons: 45
Number of swaps: 14
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 3.3:
Given an unsorted array of positive integers, design an algorithm and implement it using a
program to find whether there are any duplicate elements in the array or not. (use sorting)
(Time Complexity = O(n log n))

*/

#include <iostream>
using namespace std;

void merge(int arr[], int p, int q, int r)

{

  int n1 = q - p + 1;
  int n2 = r - q;

  int L[n1], M[n2];



  for (int i = 0; i < n1; i++)
    L[i] = arr[p + i];
  for (int j = 0; j < n2; j++)
    M[j] = arr[q + 1 + j];
  int i, j, k;
  i = 0;
  j = 0;

  k = p;

  while (i < n1 && j < n2)

  {

    if (L[i] <= M[j])

    {

      arr[k] = L[i];
      i++;
    } else

    {

      arr[k] = M[j];
      j++;
    }

    k++;

  }

  while (i < n1)

  {

    arr[k] = L[i];
    i++;
    k++;

  }



  while (j < n2)

  {

    arr[k] = M[j];
    j++;
    k++;

  }

}



void mergeSort(int arr[], int l, int r)

{

  if (l < r)

  {

    int m = l + (r - l) / 2;
    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);
    merge(arr, l, m, r);
  }

}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
void Dup()
{
    int size;
    int flag =0;
    int i;
    cout<<"Enter size of array::"<<endl;
    cin>>size;
    int arr[size];
    cout<<"Enter element::"<<endl;
    for(i=0;i<size;i++)
    {
        cin>>arr[i];
    }
            mergeSort(arr,0,size-1);
    for(i=0;i<size;i++)
    {
        if(arr[i]== arr[i+1] && i+1<size)
        {
            flag = 1;
        }
    }
    if(flag == 0)
    {
        cout<<"NO"<<endl;
    }
    else
    {
        cout<<"YES"<<endl;
    }
    return;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
int main()

{

    int testcases;

    cout<<"Number of test cases::"<<endl;
    cin>>testcases;
    for(int i=0;i<testcases;i++)

    {

        cout<<"Testcase "<<i+1<<endl;
        Dup();
    }



}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Number of test cases: 2
Test case 1:
Enter size of array: 5
Enter elements: 1 2 3 4 5
NO
Test case 2:

Enter size of array: 6
Enter elements: 1 2 3 3 4 5
YES
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

WEEK 4

```
/*
QUESTION 4.1:
Given an unsorted array of integers, design an algorithm and implement it using a program to
sort an array of elements by dividing the array into two subarrays and combining these
subarrays after sorting each one of them. Your program should also find number of
comparisons and inversions during sorting the array.

*/

#include <iostream>
 using namespace std;


int com=0;
int in=0;


void Merge(int a[], int low, int high, int mid)

{

    int i, j, k, temp[high-low+1];
    i = low;
    k = 0;

    j = mid + 1;

    while (i <= mid && j <= high)

    {

        if (a[i] < a[j])

        {

            temp[k] = a[i];
            k++;
            i++;

        }

        else

        {
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
            temp[k] = a[j];
            k++;
            j++;

            in=in+((mid-i)+1);

        }

        com++;

    }

    while (i <= mid)

    {

        temp[k] = a[i];
        k++;
        i++;

    }

    while (j <= high)

    {

        temp[k] = a[j];
        k++;
        j++;

    }

    for (i = low; i <= high; i++)

    {

        a[i] = temp[i-low];

    }

}



void MergeSort(int a[], int low, int high)

{

    int mid;

    if (low < high)

    {

        mid=(low+high)/2;
        MergeSort(a, low, mid);
        MergeSort(a, mid+1, high);
        Merge(a, low, high, mid);
    }
}



void sort()
{
    int size;
    int i;
    cout<<"Enter the size of the array::"<<endl;
    cin>>size;
    int arr[size];
    cout<<"Enter the elements::"<<endl;
    for(i=0;i<size;i++)
    {
        cin>>arr[i];
    }
    MergeSort(arr,0,size-1);
    cout<<"Elements after sorting::"<<endl;
    for(i=0;i<size;i++)
    {
        cout<<arr[i]<<"  ";
    }
    cout<<endl<<"Number of comparisons::"<<com<<endl;
    cout<<"Number of inversions::"<<in<<endl;
    return ;


}


int main()
{
    int testcases,i;
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
    cout<<"Enter the number of testcases::"<<endl;
    cin>>testcases;
    for(i=0;i<testcases;i++)
    {
        cout<<endl<<"Testcases"<<i+1<<endl;
        sort();
    }
    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Number of test cases: 2
Test case 1:
Enter size of array: 8

Enter elements: -23 65 -31 76 46 89 45 32

Elements after sorting: -31 -23 32 45 46 65 76 89
Number of comparisons: 16
Number of inversions: 13
Test case 2:
Enter size of array: 10

Enter elements: 54 65 34 76 78 97 46 32 51 21

Elements after sorting: 21 32 34 46 51 54 65 76 78 97
Number of comparisons: 22
Number of inversions: 38
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 4.2:
Given an unsorted array of integers, design an algorithm and implement it using a program to
sort an array of elements by partitioning the array into two subarrays based on a pivot
element such that one of the sub array holds values smaller than the pivot element while
another sub array holds values greater than the pivot element. Pivot element should be
selected randomly from the array.

*/

#include <iostream>
#include <cstdlib>
using namespace std;
int comparisons = 0;
int swaps = 0;

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

{

   int pivotIndex = rand() % (high - low + 1) + low;
   int  pivot = arr[pivotIndex];
   swap(arr[pivotIndex], arr[high]);
   int i = low - 1;

   for (int j = low; j < high; ++j)

   {

      if (arr[j] < pivot)

      {

         ++i;

         swap(arr[i], arr[j]);

         ++swaps;

      }

      ++comparisons;

   }
```

```
    swap(arr[i + 1], arr[high]);

    ++swaps;
    return i + 1;
}



void quickSort(int arr[], int low, int high)

{

    if (low < high)

    {

        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }

}



void sort()

{

    int size;
    int i;
    cout << "Enter the size of the array::" << endl;
    cin >> size;
    int arr[size];

    cout << "Enter the elements::" << endl;
    for (i = 0; i < size; i++)
    {

        cin >> arr[i];

    }

    quickSort(arr, 0, size - 1);

    cout << "Elements after sorting::" << endl;
    for (i = 0; i < size; i++)
    {

        cout << arr[i] << "  ";
    }

    cout << endl << "Number of comparisons::" << comparisons << endl;
    cout << "Number of swaps::" << swaps << endl;
    return;

}



int main()

{

    int testcases;

    cout << "Enter the number of testcases::" << endl;
    cin >> testcases;

    for (int i = 0; i < testcases; i++)

    {

        cout << endl << "Testcase " << i + 1 << endl;
        sort();
    }

return 0;

}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Number of test cases: 2
Test case 1:
Enter size of array: 8

Enter elements: -23 65 -31 76 46 89 45 32

Elements after sorting: -31 -23 32 45 46 65 76 89
Number of comparisons: 14
Number of swaps: 10
Test case 2:
Enter size of array: 10

Enter elements: 54 65 34 76 78 97 46 32 51 21

Elements after sorting: 21 32 34 46 51 54 65 76 78 97
Number of comparisons: 29
Number of swaps: 21
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 4.3:
Given an unsorted array of integers, design an algorithm and implement it using a program to
find Kth smallest or largest element in the array. (Worst case Time Complexity = O(n))
*/
#include <iostream>
using namespace std;
int kthSmallest(int arr[], int n, int k)
{
        int max_element = arr[0];
        for (int i = 1; i < n; i++)
   {
                if (arr[i] > max_element)
      {
                        max_element = arr[i];
                }
        }
        int freq[max_element + 1] = {0};
        for (int i = 0; i < n; i++)
   {
                freq[arr[i]]++;
        }
        int count = 0;
        for (int i = 0; i <= max_element; i++)
   {
                if (freq[i] != 0)
      {
                        count += freq[i];
                        if (count >= k)
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
            {
                                    return i;
                              }
                    }
            }
            return -1;
 }


 void sort()
 {
    int size;
    int i,k,x;
    cout << "Enter the size of the array::" << endl;
    cin >> size;
    int arr[size];
    cout << "Enter the elements::" << endl;
    for (i = 0; i < size; i++)
    {
       cin >> arr[i];
    }
    cout << "Enter the k::" << endl;
    cin >> k;
    x = kthSmallest(arr,size,k);
    if(x==-1)
    {
       cout<<"Error"<<endl;
       return;
    }
    cout << endl << "Kth smallest element::" << x << endl;
    return;
 }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
int main()

{

    int testcases;

    cout << "Enter the number of testcases::" << endl;
    cin >> testcases;


    for (int i = 0; i < testcases; i++)

    {

        cout << endl << "Testcase " << i + 1 << endl;
        sort();
    }

return 0;

}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter the number of testcases::
2
Testcase 1

Enter the size of the array::
5
Enter the elements::
7
2

3

8

1

Enter the k::

3

Kth smallest element::3
Testcase 2
Enter the size of the array::
4
Enter the elements::
4
9

5

1

Enter the k::

2

Kth smallest element::4
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

WEEK 5

```
/*
QUESTION 5.1:
Given an unsorted array of alphabets containing duplicate elements. Design an algorithm and
implement it using a program to find which alphabet has maximum number of occurrences
andprint it. (Time Complexity = O(n)) (Hint: Use counting sort)

*/

#include <bits/stdc++.h>
using namespace std;
void count()
{

   int size,i;

   cout<<"Enter the number of alphabet::"<<endl;
   cin>>size;
   char str[size];
   char ch;
   cout<<"Enter the alphabets::"<<endl;
   for ( i = 0; i < size; i++)
  {

     cin>>str[i];

  }

  int freq[26] = { 0 };
  int maxf = -1;
  for ( i = 0; i < size; i++)

  {

     freq[str[i] - 'a']++;

  }

  for ( i = 0; i < 26; i++)

  {

    if (maxf < freq[i])
```

```
        {

           maxf = freq[i];

        }

    }

    if(maxf == 0)

    {

        cout<<"No duplicate present"<<endl;
        return;
    }

    cout<<"Maximum number of frequency is of::";
    for ( i = 0; i < 26; i++)
    {

       if (freq[i]==maxf)

       {

           ch= (char)(i + 'a');
           cout<<endl<<ch<<endl;
           cout<<"The frequency of character is::"<<endl;
           cout<<freq[i];


       }

    }

 }

 int main()

 {

    int Testcase,i;

    cout<<"Enter the number of testcases::"<<endl;
    cin >>Testcase ;
    for(i=0;i<Testcase;i++)

    {

        cout<<endl<<"Testcase "<<i+1<<endl;
        count();
    }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
    return 0;
  }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter the number of testcases::
2
Testcase 1

Enter the number of alphabet::
6
Enter the alphabets::

a
b
c
d
e
f
No duplicate present


Testcase 2

Enter the number of alphabet::
7
Enter the alphabets::

a
a
a
a
a
b
c
Maximum number of frequency is of::
a
The frequency of character is::
5
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 5.2:
Given an unsorted array of integers, design an algorithm and implement it using a program to
find whether two elements exist such that their sum is equal to the given key element. (Time
Complexity = O(n log n))

*/

#include <bits/stdc++.h>
using namespace std;
void count()
{

    int size,i,j,k,x=0,sum;

    cout<<"Enter the number of element::"<<endl;
    cin>>size;
    int arr[size];

    cout<<"Enter the elements::"<<endl;
    for ( i = 0; i < size; i++)
   {

      cin>>arr[i];

   }

   cout<<"Enter sum::"<<endl;
    cin>>k;
   sort(arr,arr+size);
   i=0;
   j=size-1;

   cout<<"Answer ::"<<endl;
   while(i<j)
   {

      sum=arr[i]+arr[j];
      if(sum==k)
      {
         cout<<arr[i]<<" "<<arr[j]<<endl;
         x++;
         i++;
         j--;
      }

      else if(sum>k)

      {

          j--;

      }

      else

      {

          i++;

      }

   }

   if(x==0)

   {

      cout<<"No pair found"<<endl;

   }

   return;

}

int main()

{

   int Testcase,i;

   cout<<"Enter the number of testcases::"<<endl;
   cin >>Testcase ;
   for(i=0;i<Testcase;i++)

   {

      cout<<endl<<"Testcase "<<i+1<<endl;
      count();
   }

   return 0;

}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter the number of testcases::
2
Testcase 1

Enter the number of element::
8
Enter the elements::

-1

2

3

5

7

8

9

12

Enter sum::

10

Answer ::

-1 11

2 8

3 7

Testcase 2

Enter the number of element::
5
Enter the elements::
1
2

3

4

5

Enter sum::
10
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
No pair found
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 5.3:
You have been given two sorted integer arrays of size m and n. Design an algorithm and
implement it using a program to find list of elements which are common to both. (Time
Complexity = O(m+n))

*/

#include <iostream>
#include <algorithm>

using namespace std;


void findelements(int arr1[], int size1, int arr2[], int size2)

{

    int i = 0, j = 0;


    while (i < size1 && j < size2)

    {

        if (arr1[i] == arr2[j])

        {

            cout << arr1[i] << " ";

            ++i;

            ++j;

        }

        else if (arr1[i] < arr2[j])

        {

            ++i;

        }

        else

        {

            ++j;
        }

    }

}



void find()

{

        int size1, size2;

        cout << "Enter the size of the first array::"<<endl;
        cin >> size1;
        int arr1[size1];

        cout << "Enter elements for the first array::"<<endl;
        for (int i = 0; i < size1; ++i)
        {

            cin >> arr1[i];

        }



        cout << "Enter the size of the second array::" <<endl;
        cin >> size2;
        int arr2[size2];

        cout << "Enter elements for the second array:;"<<endl;
        for (int i = 0; i < size2; ++i)
        {

            cin >> arr2[i];

        }



        cout << "Common elements are::"<<endl;
        findelements(arr1, size1, arr2, size2);
        cout << endl;
        return ;

}



int main()
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
 {

    int testCases;

    cout << "Enter the number of test cases::"<<endl;
    cin >> testCases;


    for (int t = 0; t < testCases; ++t)

    {

      cout<<"TestCase::"<<t+1<<endl;
      find();
    }

    return 0;

 }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter the number of test cases::
2
TestCase::1

Enter the size of the first array::
6
Enter elements for the first array::
1
2

3

4

5

6

Enter the size of the second array::
5
Enter elements for the second array::
2
3

5

7

9

Common elements are::

2 3 5

TestCase::2

Enter the size of the first array::
4
Enter elements for the first array::
3
6

9

12

Enter the size of the second array::
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
3

Enter elements for the second array::
4
8

12

Common elements are::
12
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

WEEK 6

/*

Question 6.1:

Given a (directed/undirected) graph, design an algorithm and implement it using a program to find if a path exists between two given vertices or not. (Hint: use DFS)

Input Format:

Input will be the graph in the form of adjacency matrix or adjacency list.
Source vertex number and destination vertex number is also provided as an input.
Output Format:
Output will be '**Yes Path Exists**' if path exists, otherwise print '**No Such Path Exists**'.

*/

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

bool dfs(vector<vector < int>> &graph, int src, int dest, vector< bool > &visited)

{

        if (src == dest)

                return true;



        visited[src] = true;



        for (int i = 0; i < graph[src].size(); ++i)

        {

                if (graph[src][i] && !visited[i])

                {

                        if (dfs(graph, i, dest, visited))
                                return true;
                }

        }
        return false;

 }



 string isPathExists(vector<vector < int>> &graph, int src, int dest, int vertices)

 {

        vector<bool> visited(vertices, false);


        if (dfs(graph, src, dest, visited))
                return "Yes Path Exists";
        else
                return "No Such Path Exists";

 }



 int main()

 {

        int vertices, src, dest;

        cout << "Enter the number of vertices: ";
        cin >> vertices;

        vector<vector < int>> graph(vertices, vector<int> (vertices, 0));
        cout << "Enter the adjacency matrix:\n";
        for (int i = 0; i < vertices; ++i)

        {

                for (int j = 0; j < vertices; ++j)

                {

                        cin >> graph[i][j];

                }

        }



        cout << "Enter the source vertex: ";
        cin >> src;
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
        cout << "Enter the destination vertex: ";
        cin >> dest;

        cout << isPathExists(graph, src, dest, vertices) << endl;



        return 0;
 }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter the number of vertices: 4
Enter the adjacency matrix:
0 1 0 0

0 0 1 0

0 0 0 1

0 0 0 0

Enter the source vertex: 0
Enter the destination vertex: 3
Yes Path Exists
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 6.2
Given a graph, design an algorithm and implement it using a program to find if
a graph is bipartite or not. (Hint: use BFS)

Input Format:

Input will be the graph in the form of adjacency matrix or adjacency list.
Output Format:
Output will be 'Yes Bipartite' if graph is bipartite, otherwise print 'Not Bipartite'.

*/

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

bool isBipartite(vector<vector < int>> &graph, int vertices)

{

        vector<int> color(vertices, -1);


        for (int i = 0; i < vertices; ++i)

        {

                if (color[i] == -1)

                {

                        queue<int> q;
                        q.push(i);
                        color[i] = 0;

                        while (!q.empty())

                        {

                                int u = q.front();
                                q.pop();
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
                        for (int v = 0; v < vertices; ++v)
                        {
                            if (graph[u][v])
                            {
                                if (color[v] == -1)
                                {
                                    color[v] = 1 - color[u];
                                    q.push(v);
                                }
                                else if (color[v] == color[u])
                                {
                                    return false;
                                }
                            }
                        }
                    }
                }
            }

            return true;
}


int main()
{
    int vertices;
    cout << "Enter the number of vertices: ";
    cin >> vertices;

    vector<vector < int>> graph(vertices, vector<int> (vertices));
    cout << "Enter the adjacency matrix:\n";
    for (int i = 0; i < vertices; ++i)
```

```
        {

            for (int j = 0; j < vertices; ++j)

            {

                cin >> graph[i][j];

            }

        }



        if (isBipartite(graph, vertices))
            cout << "Yes Bipartite\n";
        else

            cout << "Not Bipartite\n";



        return 0;

 }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter the number of vertices: 4
Enter the adjacency matrix:
0 1 1 0

1 0 0 1

1 0 0 1

0 1 1 0

Not Bipartite

Enter the number of vertices: 3
Enter the adjacency matrix:
0 1 0

1 0 1

0 1 0

Yes Bipartite
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 6.3
Given a directed graph, design an algorithm and implement it using a program to find whether
cycle exists in the graph or not.

Input Format:

Input will be the graph in the form of adjacency matrix or adjacency list.
Output Format:
Output will be 'Yes Cycle Exists' if cycle exists otherwise print 'No Cycle Exists'.

*/

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

bool dfs(vector<vector < int>> &graph, vector< bool > &visited, vector< bool > &recStack,
int v)

{

        if (!visited[v])

        {

                visited[v] = true;
                recStack[v] = true;

                for (int i = 0; i < graph[v].size(); ++i)

                {

                        if (!visited[i] && dfs(graph, visited, recStack, i))
                                return true;
                        else if (recStack[i])

                                return true;

                }

        }
        recStack[v] = false;
        return false;
}


bool hasCycle(vector<vector < int>> &graph)

{

        int V = graph.size();
        vector<bool> visited(V, false);
        vector<bool> recStack(V, false);

        for (int i = 0; i < V; ++i)

                if (dfs(graph, visited, recStack, i))
                        return true;


        return false;

}


int main()

{

        int vertices;

        cout << "Enter the number of vertices: ";
        cin >> vertices;

        vector<vector < int>> graph(vertices, vector<int> (vertices));
        cout << "Enter the adjacency matrix:\n";
        for (int i = 0; i < vertices; ++i)

        {

                for (int j = 0; j < vertices; ++j)

                {

                        cin >> graph[i][j];

                }

        }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
        if (hasCycle(graph))
                cout << "Yes Cycle Exists\n";
        else
                cout << "No Cycle Exists\n";


        return 0;
 }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter the number of vertices: 4
Enter the adjacency matrix:
0 1 0 0

0 0 1 0

0 0 0 1

1 0 0 0

Yes Cycle Exists

Enter the number of vertices: 4
Enter the adjacency matrix:
0 1 0 0

0 0 1 0

0 0 0 0

0 0 1 0

No Cycle Exists
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

WEEK 7

```
/*
QUESTION 7.1:
After end term examination, Akshay wants to party with his friends. All his friends are
living as paying guest and it has been decided to first gather at Akshay's house and then
move towards party location. The problem is that no one knows the exact address of his
house in the city. Akshay as a computer science wizard knows how to apply his theory
subjects in his real life and came up with an amazing idea to help his friends. He draws
a graph by looking in to location of his house and his friends' location (as a node in the
graph) on a map. He wishes to find out shortest distance and path covering that distance
from each of his friend's location to his house and then whatsapp them this path so that
they can reach his house in minimum time. Akshay has developed the program that
implements Dijkstra's algorithm but not sure about correctness of results. Can you also
implement the same algorithm and verify the correctness of Akshay's results? (Hint:
Print shortest path and distance from friends' location to Akshay's house)

*/

#include <iostream>
#include <climits>
#include <vector>
using namespace std;
#define MAX_VERTICES 100

vector<vector<int>> graph(MAX_VERTICES, vector<int>(MAX_VERTICES));
int num_vertices;
int minDist(const vector<int>& dist, const vector<bool>& visited)

{

    int min = INT_MAX, min_index;

    for (int v = 0; v < num_vertices; v++)

    {

        if (!visited[v] && dist[v] < min)

        {

            min = dist[v];
            min_index = v;

        }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
        }

        return min_index;

}

void printPath(const vector<int>& parent, int v)

{

        if (parent[v] == -1)

        {

                cout << v << " ";
                 return;
        }

         printPath(parent, parent[v]);
         cout << v << " ";
}

void Print(const vector<int>& dist, const vector<int>& parent, int src, int dest)

{

        cout << "Shortest distance from " << src << " to " << dest << ": " << dist[dest]
<< endl;

        cout << "Shortest path: ";
        printPath(parent, dest);
        cout << endl;
}

void dijkstra(int src, int dest)

{

        vector<int> dist(MAX_VERTICES, INT_MAX);
        vector<bool> visited(MAX_VERTICES, false);
        vector<int> parent(MAX_VERTICES, -1);
        dist[src] = 0;

        parent[src] = -1;

        for (int count = 0; count < num_vertices - 1; count++)

        {

                int u = minDist(dist, visited);
                visited[u] = true;
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
                for (int v = 0; v < num_vertices; v++)

                {

                    if (!visited[v] && graph[u][v] && dist[u] != INT_MAX &&
                    dist[u] + graph[u][v] < dist[v])

                    {

                        dist[v] = dist[u] + graph[u][v];
                        parent[v] = u;
                    }

                }

        }

        Print(dist, parent, src, dest);

 }

 int main()

 {

        int num_friends;

        vector<int> friend_locations(MAX_VERTICES);

        cout << "Enter the number of vertices (including Akshay's house): ";
        cin >> num_vertices;
        cout << "Enter the adjacency matrix:\n";
         for (int i = 0; i < num_vertices; i++)
        {

                for (int j = 0; j < num_vertices; j++)

                {

                    cin >> graph[i][j];

                }

        }

        cout << "Enter the number of friends (excluding Akshay): ";
        cin >> num_friends;
        cout << "Enter the locations of the friends: ";
        for (int i = 0; i < num_friends; i++)
        {

                cin >> friend_locations[i];
        }

        int akshay_house = 0;

        for (int i = 0; i < num_friends; i++)

        {

                dijkstra(friend_locations[i], akshay_house);

        }

        return 0;

 }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter the number of vertices (including Akshay's house): 4
Enter the adjacency matrix:
4 2 1 4

3 0 2 1

0 5 7 6

1 0 2 0

Enter the number of friends (excluding Akshay): 2
Enter the locations of the friends: 1 4 6
Shortest distance from 1 to 0: 2

Shortest path: 1 3 0
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 7.2:
Design an algorithm and implement it using a program to solve previous question's
problem using Bellman- Ford's shortest path algorithm.

Input Format:

Input will be the graph in the form of adjacency matrix or adjacency list. Source vertex
number is also provided as an input.

Output Format:

Output will contain V lines. Each line will represent the whole path from destination
vertex number to source vertex number along with minimum path weigth.

*/



#include <iostream>
#include <climits>
#define MAX_VERTICES 100

#define MAX_EDGES 100
using namespace  std;
struct Edge

{

        int source;

        int destination;
        int weight;
};

struct Edge edges[MAX_EDGES];
int num_vertices;
int num_edges;

void printPath(int parent[], int v)

{

        if (parent[v] == -1)

        {
            cout << v << " ";
            return;
        }

        printPath(parent, parent[v]);
        cout << v << " ";
}

void printSolution(int dist[], int parent[], int src, int dest)

{

        cout << "Shortest distance from " << src << " to " << dest << ": " << dist[dest]
<< endl;

        cout << "Shortest path: ";
        printPath(parent, dest);
        cout << endl;
}

void bellmanFord(int src, int dest)

{

        int dist[MAX_VERTICES];
        int parent[MAX_VERTICES];
        for (int i = 0; i < num_vertices; i++)

        {

            dist[i] = INT_MAX;
            parent[i] = -1;
        }

        dist[src] = 0;

        for (int i = 1; i <= num_vertices - 1; i++)

        {

            for (int j = 0; j < num_edges; j++)

            {

                int u = edges[j].source;

                int v = edges[j].destination;
                int weight = edges[j].weight;
                if (dist[u] != INT_MAX && dist[u] + weight < dist[v])
                {
                    dist[v] = dist[u] + weight;
                    parent[v] = u;
                }
            }
        }
        for (int j = 0; j < num_edges; j++)
        {
                int u = edges[j].source;
                int v = edges[j].destination;
                int weight = edges[j].weight;
                if (dist[u] != INT_MAX && dist[u] + weight < dist[v])
                {
                    cout << "Negative weight cycle." << endl;
                    return;
                }
        }
        printSolution(dist, parent, src, dest);
}
int main()
{
        int num_friends;
        int friend_locations[MAX_VERTICES];
        cout << "Enter the number of vertices (including Akshay's house): ";
        cin >> num_vertices;
        cout << "Enter the number of edges: ";
        cin >> num_edges;
        cout << "Enter the edges (source destination weight):\n";
        for (int i = 0; i < num_edges; i++)
        {
            cin >> edges[i].source >> edges[i].destination >> edges[i].weight;
        }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
    cout << "Enter the number of friends (excluding Akshay): ";
    cin >> num_friends;
    cout << "Enter the locations of the friends (node numbers): ";
    for (int i = 0; i < num_friends; i++)
    {
        cin >> friend_locations[i];
    }
    int akshay_house = 0;
    for (int i = 0; i < num_friends; i++)
    {
        bellmanFord(friend_locations[i], akshay_house);
    }
    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter the number of vertices (including Akshay's house): 3
Enter the number of edges: 3
Enter the edges (source destination weight):
5 2 3
1 0 4

6 1 4

Enter the number of friends (excluding Akshay): 2
Enter the locations of the friends (node numbers): 1 3 4
Shortest distance from 1 to 0: 4
Shortest path: 1 0

Shortest distance from 3 to 0: 8
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 7.3:
Given a directed graph with two vertices ( source and destination). Design an algorithm
and implement it using a program to find the weight of the shortest path from source to
destination with exactly k edges on the path.

Input Format:

First input line will obtain number of vertices V present in the graph. Graph in the form
of adjacency matrix or adjacency list is taken as an input in next V line . Next input line
will obtain source and destination vertex number. Last input line will obtain value k.

Output Format:
Output will be the weigth of shortest path from source to destination having exactly k edges.
If no path is available then print "no path of length k is available".

*/

#include <iostream>
#include <climits>
using namespace std;
#define MAX_VERTICES 100

void shortestPathWithKEdges(int graph[MAX_VERTICES][MAX_VERTICES], int
numVertices, int source, int destination, int k)

{

        int dp[MAX_VERTICES][MAX_VERTICES][k + 1];

        for (int i = 1; i <= numVertices; i++)

        {

                for (int j = 1; j <= numVertices; j++)

                {

                        for (int l = 0; l <= k; l++)

                        {

                                if (l == 0)

                                {
                                        dp[i][j][l] = graph[i][j];

                                }
```

```
                    else
                    {
                        dp[i][j][l] = INT_MAX;
                    }
                }
            }
        }
        for (int l = 1; l <= k; l++)
        {
            for (int x = 1; x <= numVertices; x++)
            {
                for (int i = 1; i <= numVertices; i++)
                {
                    for (int j = 1; j <= numVertices; j++)
                    {
                        if (dp[i][x][l - 1] != INT_MAX
                        && dp[x][j][1] != INT_MAX)
                        {
                            dp[i][j][l] = min(dp[i][j][l],
                           dp[i][x][l - 1] +
                           dp[x][j][1]);
                        }
                    }
                }
            }
        }
        int shortestPathWeight = dp[source][destination][k];
        if (shortestPathWeight == INT_MAX)
        {
            cout << "No path found with exactly " << k << " edges from source to
            destination." << endl;
        }
        else
        {
              cout << "Weight of the shortest path with exactly " << k << " edges
              from source to destination: " << shortestPathWeight << endl;
        }
}


int main()
{
        int numVertices;
        cout << "Enter the number of vertices in the graph: ";
        cin >> numVertices;
        int graph[MAX_VERTICES][MAX_VERTICES];
        cout << "Enter the adjacency matrix representing the graph:" << endl;
        for (int i = 1; i <= numVertices; i++)
        {
                for (int j = 1; j <= numVertices; j++)
                {
                        cin >> graph[i][j];
                }
        }
        int source, destination, k;
        cout << "Enter the source vertex: ";
        cin >> source;
        cout << "Enter the destination vertex: ";
        cin >> destination;
        cout << "Enter the number of edges allowed on the path (k): ";
        cin >> k;
        shortestPathWithKEdges(graph, numVertices, source, destination, k);
        return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter the number of vertices in the graph: 4

Enter the adjacency matrix representing the graph:

1 0 2 0

3 1 6 0

0 0 9 4

5 6 2 7

Enter the source vertex: 2
Enter the destination
vertex: 9
Enter the number of edges allowed on the path (k): 5 4

Weight of the shortest path with exactly 4 edges from source to destination: 0
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

WEEK 8

```
/*
QUESTION 8.1:

Assume that a project of road construction to connect some cities is given to your friend.
Map of these cities and roads which will connect them (after construction) is provided to
him in the form of a graph. Certain amount of rupees is associated with construction of each
road. Your friend has to calculate the minimum budget required for this project. The budget
should be designed in such a way that the cost of connecting the cities should be minimum
and number of roads required to connect all the cities should be minimum (if there are N
cities then only N-1 roads need to be constructed). He asks you for help. Now, you have to
help your friend by designing an algorithm which will find minimum cost required to
connect these cities. (use Prim's algorithm)

*/

#include <iostream>

#include <vector>

#include <queue>

#include <climits>

using namespace std;


int primMST(vector<vector < int>> &graph)

{

	int  V = graph.size();

	vector<int> key(V, INT_MAX);

	vector<int> parent(V, -1);

	vector<bool> inMST(V, false);


	priority_queue<pair<int, int>, vector< pair<int, int>>, greater<pair<int, int>>> pq;


	pq.push({ 0, 0 });

	key[0] = 0;
```

```
        while (!pq.empty())
        {
                int u = pq.top().second;
                pq.pop();

                inMST[u] = true;

                for (int v = 0; v < V; ++v)
                {
                        if (!inMST[v] && graph[u][v] && graph[u][v] < key[v])
                        {
                                key[v] = graph[u][v];
                                parent[v] = u;
                                pq.push({ key[v], v });
                        }
                }
        }

        int minCost = 0;
        for (int i = 1; i < V; ++i)
                minCost += key[i];

        return minCost;
  }

  int main()
  {
        int numCities;
        cout << "Enter the number of cities: ";
        cin >> numCities;
        vector<vector < int>> graph(numCities, vector<int> (numCities));
        cout << "Enter the adjacency matrix representing the graph :"
        for (int i = 0; i < numCities; ++i)
        {
                for (int j = 0; j < numCities; ++j)
                {
                        cin >> graph[i][j];
                }
        }

        int minCost = primMST(graph);
        cout << "Minimum cost to connect all cities: " << minCost << endl;

        return 0;
  }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter the number of cities: 4

Enter the adjacency matrix representing the graph :

0 10 6 5

10 0 0 15

6 0 0 4

5 15 4 0

Minimum cost to connect all cities: 19


Enter the number of cities: 3

Enter the adjacency matrix representing the graph :

0 2 0

2 0 3

0 3 0

Minimum cost to connect all cities: 5
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 8.2:

Implement the previous problem using Kruskal's algorithm.

Input Format:

The first line of input takes number of vertices in the graph.

Input will be the graph in the form of adjacency matrix or adjacency list.

Output Format:

Output will be minimum spanning weight

*/

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;


struct Edge

{

	int src, dest, weight;

};


struct Subset

{

	int parent, rank;

};


bool compareEdges(const Edge &a, const Edge &b)

{

	return a.weight < b.weight;
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
}


int find(vector<Subset> &subsets, int i)
{
        if (subsets[i].parent != i)
                subsets[i].parent = find(subsets, subsets[i].parent);
        return subsets[i].parent;
}

void Union(vector<Subset> &subsets, int x, int y)
{
        int xroot = find(subsets, x);
        int yroot = find(subsets, y);

        if (subsets[xroot].rank < subsets[yroot].rank)
                subsets[xroot].parent = yroot;
        else if (subsets[xroot].rank > subsets[yroot].rank)
                subsets[yroot].parent = xroot;
        else
        {
                subsets[yroot].parent = xroot;
                subsets[xroot].rank++;
        }
}


int kruskalMST(vector<vector < int>> &graph, int V)
{
        vector<Edge> edges;
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
    for (int i = 0; i < V; ++i)
    {
        for (int j = i + 1; j < V; ++j)
        {
            if (graph[i][j] != 0)
            {
                Edge edge;
                edge.src = i;
                edge.dest = j;
                edge.weight = graph[i][j];
                edges.push_back(edge);
            }
        }
    }

    sort(edges.begin(), edges.end(), compareEdges);

    vector<Subset> subsets(V);

    for (int i = 0; i < V; ++i)
    {
        subsets[i].parent = i;
        subsets[i].rank = 0;
    }

    int minCost = 0;
    int e = 0;
    int i = 0;
    while (e < V - 1 && i < edges.size())
```

```
        {
            Edge next_edge = edges[i++];
            int x = find(subsets, next_edge.src);
            int y = find(subsets, next_edge.dest);

            if (x != y)
            {
                minCost += next_edge.weight;
                Union(subsets, x, y);
                ++e;
            }
        }

        return minCost;
    }

    int main()
    {
        int V;
        cin >> V;

        vector<vector < int>> graph(V, vector<int> (V));
        for (int i = 0; i < V; ++i)
        {
            for (int j = 0; j < V; ++j)
            {
                cin >> graph[i][j];
            }
        }
        int numEdges = 0;
        for (int i = 0; i < V; ++i)
        {
                for (int j = i + 1; j < V; ++j)
                {
                        if (graph[i][j] != 0)
                        {
                                numEdges++;
                        }
                }
        }

        int minCost = kruskalMST(graph, V);

        cout << "Number of vertices: " << V << endl;
        cout << "Number of edges: " << numEdges << endl;
        cout << "Minimum spanning weight: " << minCost << endl;

        return 0;
  }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
5

0 1 2 0 0

1 0 3 0 4

2 3 0 5 0

0 0 5 0 6

0 4 0 6 0

Number of vertices: 5

Number of edges: 7

Minimum spanning weight: 12
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 8.3:

Assume that same road construction project is given to another person. The amount he will
earn from this project is directly proportional to the budget of the project. This person is
greedy, so he decided to maximize the budget by constructing those roads who have highest
construction cost. Design an algorithm and implement it using a program to find the
maximum budget required for the project

*/
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

struct Edge
{
	int src, dest, weight;
};

struct Subset
{
	int parent, rank;
};

bool compareEdges(const Edge &a, const Edge &b)
{
	return a.weight > b.weight;
}

int find(vector<Subset> &subsets, int i)
{
	if (subsets[i].parent != i)
		subsets[i].parent = find(subsets, subsets[i].parent);
	return subsets[i].parent;
}

void Union(vector<Subset> &subsets, int x, int y)
{
	int xroot = find(subsets, x);
	int yroot = find(subsets, y);
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
        if (subsets[xroot].rank < subsets[yroot].rank)
                subsets[xroot].parent = yroot;
        else if (subsets[xroot].rank > subsets[yroot].rank)
                subsets[yroot].parent = xroot;
        else
        {
                subsets[yroot].parent = xroot;
                subsets[xroot].rank++;
        }
}


int maxBudgetMST(vector<vector < int>> &graph, int V)
{
        vector<Edge> edges;

        for (int i = 0; i < V; ++i)
        {
                for (int j = i + 1; j < V; ++j)
                {
                        if (graph[i][j] != 0)
                        {
                                Edge edge;
                                edge.src = i;
                                edge.dest = j;
                                edge.weight = graph[i][j];
                                edges.push_back(edge);
                        }
                }
        }

        sort(edges.begin(), edges.end(), compareEdges);

        vector<Subset> subsets(V);

        for (int i = 0; i < V; ++i)
        {
                subsets[i].parent = i;
                subsets[i].rank = 0;
        }

        int maxBudget = 0;
        for (const Edge &edge: edges)
        {
            int x = find(subsets, edge.src);
            int y = find(subsets, edge.dest);

            if (x != y)
            {
                maxBudget += edge.weight;
                Union(subsets, x, y);
            }
        }

        return maxBudget;
}

int main()
{
        int V;
        cin >> V;

        vector<vector < int>> graph(V, vector<int> (V));
        for (int i = 0; i < V; ++i)
        {
            for (int j = 0; j < V; ++j)
            {
                cin >> graph[i][j];
            }
        }

        int maxBudget = maxBudgetMST(graph, V);

        cout << maxBudget << endl;

        return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT7

```
0 0 7 5 0 0 0

0 0 8 5 0 0 0

7 8 0 9 7 0 0

5 0 9 0 1 5 6

0 5 7 1 5 0 8

0 0 0 6 8 0 11

0 0 0 0 9 11 0

59
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

WEEK9

/*

QUESTION 9.1-

Given a graph, Design an algorithm and implement it using a program to implement Floyd-Warshall all pair shortest path algorithm. Input Format: The first line of input takes number of vertices in the graph. Input will be the graph in the form of adjacency matrix or adjacency list. If a direct edge is not present between any pair of vertex (u,v), then this entry is shown as AdjM[u,v] = INF. Output Format: Output will be shortest distance matrix in the form of V X V matrix, where each entry (u,v) represents shortest distance between vertex u and vertex v

```
#include <iostream>

#include <vector>

#include <climits>

#define INF INT_MAX

using namespace std;


void floydWarshall(vector<vector < int>> &graph, int V)

{

    vector<vector < int>> dist = graph;


    for (int k = 0; k < V; k++)

    {

        for (int i = 0; i < V; i++)

        {

            for (int j = 0; j < V; j++)

            {

                if (dist[i][k] != INF && dist[k][j] != INF && dist[i][k] + dist[k][j] <
dist[i][j])

                {

                    dist[i][j] = dist[i][k] + dist[k][j];

                }

            }
        }
    }

    for (int i = 0; i < V; i++)
    {
        for (int j = 0; j < V; j++)
        {
            if (dist[i][j] == INF)
            {
                cout << "INF ";
            }
            else
            {
                cout << dist[i][j] << " ";
            }
        }

        cout << endl;
    }
}

int main()
{
    int V;
    cout << "Enter the number of vertices: ";
    cin >> V;

    vector<vector < int>> graph(V, vector<int> (V));
    cout << "Enter the adjacency matrix (Enter INF if there's no direct edge):\n";
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
    for (int i = 0; i < V; i++)
    {
        for (int j = 0; j < V; j++)
        {
            cin >> graph[i][j];
        }
    }

    cout << "Shortest distance matrix:\n";
    floydWarshall(graph, V);

    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter the number of vertices: 4

Enter the adjacency matrix (Enter INF if there's no direct edge):

0 5 INF 10

INF 0 3 INF

INF INF 0 1

INF INF INF 0

Shortest distance matrix:

0 5 8 9

INF 0 3 4

INF INF 0 1

INF INF INF 0
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

/*

QUESTION 9.2-

Given a knapsack of maximum capacity w. N items are provided, each having its own value and weight. You have to Design an algorithm and implement it using a program to find the list of the selected items such that the final selected content has weight w and has maximum value. You can take fractions of items,i.e. the items can be broken into smaller pieces so that you have to carry only a fraction xi of item i, where 0 ≤xi≤ 1.

Input Format: First input line will take number of items N which are provided. Second input line will contain N space-separated array containing weights of all N items. Third input line will contain N space-separated array containing values of all N items. Last line of the input will take the maximum capacity w of knapsack.

Output Format: First output line will give maximum value that can be achieved. Next Line of output will give list of items selected along with their fraction of amount which has been taken. Sample I/O Problem

```
#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;


struct Item

{

        int weight;

        int value;

        double ratio;

        int index;

};


bool compare(Item a, Item b)

{

        return a.ratio > b.ratio;

}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
void fractionalKnapsack(int N, vector<int> &weights, vector<int> &values, int capacity)
{
    vector<Item> items(N);

    for (int i = 0; i < N; i++)
    {
        items[i].weight = weights[i];
        items[i].value = values[i];
        items[i].ratio = (double) values[i] / weights[i];
        items[i].index = i;
    }

    sort(items.begin(), items.end(), compare);

    int currentWeight = 0;
    double finalValue = 0.0;
    vector<pair<int, int>> selectedItems;

    for (int i = 0; i < N; i++)
    {
        if (currentWeight + items[i].weight <= capacity)
        {
            currentWeight += items[i].weight;
            finalValue += items[i].value;
            selectedItems.push_back({ items[i].index + 1, items[i].weight });
        }
        else
        {
            int remainingWeight = capacity - currentWeight;
            finalValue += items[i].ratio * remainingWeight;
            selectedItems.push_back({ items[i].index + 1, remainingWeight });
            break;
        }
    }

    cout << "Maximum value: " << finalValue << endl;
    cout << "item-weight" << endl;
    for (auto item: selectedItems)
    {
        cout << item.first << "-" << item.second << endl;
    }
}

int main()
{
    int N;
    cout << "Enter the number of items: ";
    cin >> N;

    vector<int> weights(N);
    vector<int> values(N);

    cout << "Enter the weights of the items: ";
    for (int i = 0; i < N; i++)
    {
        cin >> weights[i];
    }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
    cout << "Enter the values of the items: ";
    for (int i = 0; i < N; i++)
    {
        cin >> values[i];
    }

    int capacity;
    cout << "Enter the maximum capacity of the knapsack: ";
    cin >> capacity;

    fractionalKnapsack(N, weights, values, capacity);

    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter the number of items: 6

Enter the weights of the items: 5 6 4 1 3 7

Enter the values of the items: 1 3 5 6 8 10

Enter the maximum capacity of the knapsack: 15

Maximum value: 22.3333

item-weight

5-1

6-3

4-5

1-6

3-1
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*

QUESTION 9.3-

Given an array of elements. Assume arr[i] represents the size of file i. Write an algorithm and a program to merge all these files into single file with minimum computation. For given two files A and B with sizes m and n, computation cost of merging them is O(m+n). (Hint: use greedy approach)

Input Format: First line will take the size n of the array. Second line will take array s an input.

Output Format: Output will be the minimum computation cost required to merge all the elements of the array

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;


int minComputationCost(vector<int> &arr)

{

    sort(arr.begin(), arr.end());

    int total_cost = 0;

    while (arr.size() > 1)

    {

        int merge_cost = arr[0] + arr[1];

        total_cost += merge_cost;

        arr.erase(arr.begin(), arr.begin() + 2);

        auto it = lower_bound(arr.begin(), arr.end(), merge_cost);

        arr.insert(it, merge_cost);

    }


    return total_cost;

}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

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

    vector<int> arr(n);
    cout << "Enter the array of file sizes: ";
    for (int i = 0; i < n; ++i)
    {
        cin >> arr[i];
    }

    int cost = minComputationCost(arr);
    cout << "Minimum computation cost required: " << cost << endl;

    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Enter the size of the array: 10

Enter the array of file sizes: 10 5 100 50 20 15 5 20 100 10

Minimum computation cost required: 960

Enter the size of the array: 6

Enter the array of file sizes: 5 10 15 20 50 100

Minimum computation cost required: 395
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

WEEK 10

/*

QUESTION 10.1-

Given a list of activities with their starting time and finishing time. Your goal is to select maximum number of activities that can be performed by a single person such that selected activities must be non-conflicting. Any activity is said to be non-conflicting if starting time of an activity is greater than or equal to the finishing time of the other activity. Assume that a person can only work on a single activity at a time.

Input Format:

First line of input will take number of activities N.

Second line will take N space-separated values defining starting time for all the N activities.

Third line of input will take N space-separated values defining finishing time for all the N activities.

Output Format:

Output will be the number of non-conflicting activities and the list of selected activities.

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
    int N;
    cin >> N;
    vector<int> start_times(N);
    vector<int> finish_times(N);

    for (int i = 0; i < N; ++i) {
        cin >> start_times[i];
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
    }


    for (int i = 0; i < N; ++i) {
        cin >> finish_times[i];
    }


    vector<pair<int, pair<int, int>>> activities;
    for (int i = 0; i < N; ++i) {
        activities.push_back({finish_times[i], {start_times[i], i + 1}});
    }


    sort(activities.begin(), activities.end());


    vector<int> selected_activities;
    int last_finish_time = 0;


    for (const auto& activity : activities) {
        if (activity.second.first >= last_finish_time) {
            selected_activities.push_back(activity.second.second);
            last_finish_time = activity.first;
        }
    }


    cout << "No. of non-conflicting activities: " << selected_activities.size() << endl;
    cout << "List of selected activities: ";
    for (size_t i = 0; i < selected_activities.size(); ++i) {
        if (i > 0) cout << ", ";
        cout << selected_activities[i];
    }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
        cout << endl;



        return 0;
    }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT:

```
6

1 3 0 5 8 5

2 4 6 7 9 9

No. of non-conflicting activities: 4

List of selected activities: 1, 2, 4, 5
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*
QUESTION 10.2-
Given a long list of tasks. Each task takes specific time to accomplish it and each task has a
 deadline associated with it. You have to design an algorithm and implement it using a program to
 find maximum number of tasks that can be completed without crossing their deadlines and also
 find list of selected tasks.
 Input Format:
 First line will give total number of tasks n.
 Second line of input will give n space-separated elements of array representing time taken by
 each task.
 Third line of input will give n space-separated elements of array representing deadline associated
 with each task.
 Output Format:
 Output will be the total number of maximum tasks that can be completed.
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

struct Task {
    int time;
    int deadline;
    int index;
};

bool compareTasks(const Task& a, const Task& b) {
    return a.deadline < b.deadline;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
int main() {
    int n;
    cin >> n;

    vector<int> times(n);
    vector<int> deadlines(n);

    for (int i = 0; i < n; ++i) {
        cin >> times[i];
    }

    for (int i = 0; i < n; ++i) {
        cin >> deadlines[i];
    }

    vector<Task> tasks(n);
    for (int i = 0; i < n; ++i) {
        tasks[i] = {times[i], deadlines[i], i + 1};
    }

    sort(tasks.begin(), tasks.end(), compareTasks);

    vector<int> selectedTasks;
    int currentTime = 0;

    for (const Task& task : tasks) {
        if (currentTime + task.time <= task.deadline) {
            selectedTasks.push_back(task.index);
```

```
                currentTime += task.time;
            }
        }


        cout << "Max number of tasks = " << selectedTasks.size() << endl;
        cout << "Selected task numbers: ";
        for (size_t i = 0; i < selectedTasks.size(); ++i) {
            if (i > 0) cout << ", ";
            cout << selectedTasks[i];
        }
        cout << endl;


        return 0;
    }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
7

3 2 1 2 1 3 2

2 2 1 3 2 3 2

Max number of tasks = 3

Selected task numbers: 3, 1, 4
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*

QUESTION 10.3-

III. Given an unsorted array of elements, design an algorithm and implement it using a program to
 find whether majority element exists or not. Also find median of the array. A majority element is
 an element that appears more than n/2 times, where n is the size of array.

 Input Format:

 First line of input will give size n of array.

Second line of input will take n space-separated elements of array.

 Output Format:

 First line of output will be 'yes' if majority element exists, otherwise print 'no'.

 Second line of output will print median of the array.

#include <iostream>

#include <vector>

#include <algorithm>


using namespace std;


int findMajorityElement(const vector<int>& nums) {

    int candidate = -1;

    int count = 0;

    for (int num : nums) {

        if (count == 0) {

            candidate = num;

            count = 1;

        } else if (num == candidate) {

            count++;

        } else {

            count--;

        }
    }
    count = 0;
    for (int num : nums) {
        if (num == candidate) {
            count++;
        }
    }
    if (count > nums.size() / 2) {
        return candidate;
    } else {
        return -1;
    }
}

double findMedian(vector<int>& nums) {
    sort(nums.begin(), nums.end());
    int n = nums.size();
    if (n % 2 == 0) {
        return (nums[n / 2 - 1] + nums[n / 2]) / 2.0;
    } else {
        return nums[n / 2];
    }
}

int main() {
    int n;
    cin >> n;

    vector<int> nums(n);
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
    for (int i = 0; i < n; ++i) {
        cin >> nums[i];
    }

    int majorityElement = findMajorityElement(nums);
    if (majorityElement != -1) {
        cout << "yes" << endl;
    } else {
        cout << "no" << endl;
    }

    double median = findMedian(nums);
    cout << median << endl;

    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
9

2 2 1 1 1 2 2 2 2

yes

2
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

WEEK 11

/*

QUESTION 11.1-

Given a sequence of matrices, write an algorithm to find most efficient way to multiply these matrices together. To find the optimal solution, you need to find the order in which these matrices should be multiplied.

Input Format:

First line of input will take number of matrices n that you need to multiply.

For each line i in n, take two inputs which will represent dimensions aXb of matrix i.

Output Format:

Output will be the minimum number of operations that are required to multiply the list of matrices.

```
#include <iostream>
#include <vector>
#include <climits>

using namespace std;

int matrixChainOrder(const vector<int>& p, int n) {
    vector<vector<int>> m(n, vector<int>(n, 0));

    for (int L = 2; L < n; ++L) {
        for (int i = 1; i < n - L + 1; ++i) {
            int j = i + L - 1;
            m[i][j] = INT_MAX;
            for (int k = i; k <= j - 1; ++k) {
                int q = m[i][k] + m[k + 1][j] + p[i - 1] * p[k] * p[j];
                if (q < m[i][j])
                    m[i][j] = q;
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
                }
            }
        }
        return m[1][n - 1];
    }

    int main() {
        int n;
        cin >> n;
        vector<int> p(n + 1);

        for (int i = 0; i < n; ++i) {
            int r, c;
            cin >> r >> c;
            if (i == 0) {
                p[i] = r;
            }
            p[i + 1] = c;
        }

        cout << matrixChainOrder(p, n + 1) << endl;

        return 0;
    }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT:

```
3

10 30

30 5

5 60

4500
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

/*

QUESTION 11.2-

Given a set of available types of coins. Let suppose you have infinite supply of each type of coin. For a given value N, you have to Design an algorithm and implement it using a program to find number of ways in which these coins can be added to make sum value equals to N.

Input Format:

First line of input will take number of coins that

are available. Second line of input will take the

value of each coin.

Third line of input will take the value N for which you need to find sum.

Output Format:

Output will be the number of ways

```
#include <iostream>

#include <vector>


using namespace std;


int countWays(int coins[], int m, int N) {

    vector<int> dp(N + 1, 0);

    dp[0] = 1;


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

        for (int j = coins[i]; j <= N; ++j) {

            dp[j] += dp[j - coins[i]];

        }

    }


    return dp[N];
}


int main() {
    int m;
    cin >> m;
    int coins[m];


    for (int i = 0; i < m; ++i) {
        cin >> coins[i];
    }


    int N;
    cin >> N;


    cout << countWays(coins, m, N) << endl;


    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUTS

```
4

2 3 6 5

10

5
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

QUESTION 11.3-

Given a set of elements, you have to partition the set into two subsets such that the sum of elements in both subsets is same. Design an algorithm and implement it using a program to solve this problem.

Input Format:

First line of input will take number of elements n present in the set.

Second line of input will take n space-separated elements of the set.

Output Format:

Output will be 'yes' if two such subsets found otherwise print 'no'.

```
#include <iostream>
#include <vector>

using namespace std;

bool canPartition(vector<int>& nums) {
    int totalSum = 0;
    for (int num : nums) {
        totalSum += num;
    }

    if (totalSum % 2 != 0) {
        return false;
    }

    int targetSum = totalSum / 2;
    int n = nums.size();

    vector<vector<bool>> dp(n + 1, vector<bool>(targetSum + 1, false));
    for (int i = 0; i <= n; ++i) {
        dp[i][0] = true;
    }

    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= targetSum; ++j) {
            if (j >= nums[i - 1]) {
                dp[i][j] = dp[i - 1][j] || dp[i - 1][j - nums[i - 1]];
            } else {
                dp[i][j] = dp[i - 1][j];
            }
        }
    }

    return dp[n][targetSum];
}

int main() {
    int n;
    cin >> n;
    vector<int> nums(n);
    for (int i = 0; i < n; ++i) {
        cin >> nums[i];
    }

    if (canPartition(nums)) {
        cout << "yes" << endl;
    } else {
        cout << "no" << endl;
    }
    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
4

1 2 3 4

no
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

WEEK 12

```
/*

QUESTION 12.1-

Given two sequences, Design an algorithm and implement it using a program to
 find the length of longest subsequence present in both of them. A subsequence is a
 sequence that appears in the same relative order, but not necessarily contiguous.
 Input Format:
 First input line will take character
 sequence 1. Second input line will
 take character sequence 2.
 Output Format:
 Output will be the longest common subsequence along with its length.
*/
#include <iostream>
#include <cstring>
using namespace std;

int longestCommonSubsequence(string s1, string s2) {
    int n = s1.length();
    int m = s2.length();
    int dp[n+1][m+1];
    memset(dp, 0, sizeof(dp));

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            if (s1[i-1] == s2[j-1]) {
                dp[i][j] = dp[i-1][j-1] + 1;
            } else {
                dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
            }
        }
    }

    return dp[n][m];
}

int main() {
    string s1, s2;
    cout << "Enter sequence 1: ";
    cin >> s1;
    cout << "Enter sequence 2: ";
    cin >> s2;
    int length = longestCommonSubsequence(s1, s2);
    cout << "Length of longest common subsequence: " << length << endl;
    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT:

```
ABCBDAB

BDCABB

Length of longest common subsequence: 4
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*

QUESTION 12.2-

Given a knapsack of maximum capacity w. N items are provided, each having its own value and
 weight. Design an algorithm and implement it using a program to find the list of the selected
 items such that the final selected content has weight <= w and has maximum value. Here, you
 cannot break an item i.e. either pick the complete item or don't pick it. (0-1 property).
 Input Format:
 First line of input will provide number of items n.
 Second line of input will take n space-separated integers describing weights for all items.
 Third line of input will take n space-separated integers describing value for each item.
 Last line of input will give the knapsack capacity.
 Output Format:
 Output will be maximum value that can be achieved and list of items selected along with their
weight and value
*/


#include <iostream>

#include <vector>

using namespace std;


int knapsack(int W, vector<int>& weights, vector<int>& values, vector<int>& selectedItems) {

    int n = weights.size();

    vector<vector<int>> dp(n + 1, vector<int>(W + 1, 0));


    for (int i = 1; i <= n; i++) {

        for (int j = 1; j <= W; j++) {

            if (weights[i - 1] <= j) {

                dp[i][j] = max(values[i - 1] + dp[i - 1][j - weights[i - 1]], dp[i - 1][j]);

            } else {
                dp[i][j] = dp[i - 1][j];
            }
        }
    }

    int maxVal = dp[n][W];
    int w = W;
    for (int i = n; i > 0 && maxVal > 0; i--) {
        if (maxVal != dp[i - 1][w]) {
            selectedItems.push_back(i - 1);
            maxVal -= values[i - 1];
            w -= weights[i - 1];
        }
    }
    return dp[n][W];
}

int main() {
    int n, W;
    cout << "Enter number of items: ";
    cin >> n;
    vector<int> weights(n), values(n);
    cout << "Enter weights for all items: ";
    for (int i = 0; i < n; i++) {
        cin >> weights[i];
    }
    cout << "Enter values for all items: ";
    for (int i = 0; i < n; i++) {
        cin >> values[i];
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
    }
    cout << "Enter knapsack capacity: ";
    cin >> W;

    vector<int> selectedItems;
    int maxValue = knapsack(W, weights, values, selectedItems);

    cout << "Maximum value that can be achieved: " << maxValue << endl;
    cout << "Items selected (weight, value):" << endl;
    for (int i = selectedItems.size() - 1; i >= 0; i--) {
        cout << "(" << weights[selectedItems[i]] << ", " << values[selectedItems[i]] << ")" << endl;
    }

    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
Number of items: 4

Weights for all items: 2 3 4 5

Values for all items: 3 4 5 6

Knapsack capacity: 7

Maximum value that can be achieved: 10

Items selected (weight, value):

(5, 6)

(2, 3)
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*

QUESTION 12.3-

Given a string of characters, design an algorithm and implement it using a program to print all
possible permutations of the string in lexicographic order.

Input Format: String of characters is provided as an input.

Output Format: Output will be the list of all possible permutations in lexicographic order
*/


#include <iostream>

using namespace std;


void generatePermutations(string& str, int index, int length) {

    if (index == length - 1) {

        cout << str << endl;

        return;

    }


    for (int i = index; i < length; i++) {

        swap(str[index], str[i]);

        generatePermutations(str, index + 1, length);

        swap(str[index], str[i]);

    }

}


int main() {

    string s;

    cout << "Enter a string of characters: ";

    cin >> s;
```

```
        generatePermutations(s, 0, s.length());



        return 0;

    }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTOUT

```
XYZ


XYZ
XZY
YXZ
YZX
ZXY
ZYX
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

WEEK 13

```
/*

QUESTION 13.1-

Given an array of characters, you have to find distinct characters from this array. Design an algorithm and implement it using a program to solve this problem using hashing. (Time Complexity = O(n))

Input Format: First line of input will give the size n of the character array. Second line of input will give n space-separated elements to character array.

Output Format: Output will be the list of characters present in the array in alphabetical order and frequency of each character in the array
*/


#include <iostream>

#include <unordered_map>

#include <vector>

#include <algorithm>

using namespace std;


int main() {

    int n;

    cin >> n;


    vector<char> arr(n);

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

        cin >> arr[i];

    }


    unordered_map<char, int> freq;

    for (char c : arr) {

        freq[c]++;

    }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
    sort(arr.begin(), arr.end());


    cout << "Distinct characters and their frequencies:" << endl;
    for (char c : arr) {
        if (freq[c] > 0) {
            cout << c << ": " << freq[c] << endl;
            freq[c] = 0;
        }
    }


    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
8

X Y Z X Y X Z Y

Distinct characters and their frequencies:

X: 3

Y: 3

Z: 2
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*

QUESTION 13.2-

Given an array of integers of size n, design an algorithm and write a program to check whether this
array contains duplicates within a small window of size k<n.
*/


#include <iostream>

#include <unordered_set>

#include <vector>

using namespace std;


bool containsDuplicateWithinWindow(const vector<int>& nums, int k) {

    unordered_set<int> window;

    for (int i = 0; i < nums.size(); ++i) {

        if (window.count(nums[i]) > 0) {

            return true;

        }

        window.insert(nums[i]);

        if (i >= k) {

            window.erase(nums[i - k]);

        }

    }

    return false;

}


int main() {

    int T;

    cin >> T;
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
    while (T--) {
        int n, k;
        cin >> n >> k;


        vector<int> nums(n);
        for (int i = 0; i < n; ++i) {
            cin >> nums[i];
        }


        if (containsDuplicateWithinWindow(nums, k)) {
            cout << "Yes, array contains duplicate within a window of size " << k << "." << endl;
        } else {
            cout << "No, array does not contain duplicate within a window of size " << k << "." <<
endl;
        }
    }


    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
1

6 2

3 4 2 5 2 4

Yes, array contains duplicate within a window of size 2.
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

/*

QUESTION 13.3-

Given an array of non-negative integers, design an algorithm and implement it using a program to find two pairs (a, b) and (c,d)such that $a×b=c×d$, where a,b,c and d are distinct elements of the array.

Input Format:

- The first line of input will give the size of the array nn.
- The second line of input will give space-separated array elements.

Output Format:

- The first line of the output will give the pair (a,b).
- The second line of the output will give the pair (c,d).

*/

```
#include <iostream>

#include <unordered_map>

#include <vector>

using namespace std;


void findPairs(const vector<int>& nums) {

    unordered_map<int, pair<int, int>> productMap;

    int n = nums.size();


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

        for (int j = i + 1; j < n; ++j) {

            int product = nums[i] * nums[j];

            if (productMap.find(product) == productMap.end()) {

                productMap[product] = make_pair(nums[i], nums[j]);

            } else {

                pair<int, int> firstPair = productMap[product];
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
                cout << "(" << firstPair.first << ", " << firstPair.second << ")" << endl;
                cout << "(" << nums[i] << ", " << nums[j] << ")" << endl;
                return;
            }
        }
    }

    cout << "No such pairs found." << endl;
}

int main() {
    int n;
    cin >> n;

    vector<int> nums(n);
    for (int i = 0; i < n; ++i) {
        cin >> nums[i];
    }

    findPairs(nums);

    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

```
5

1 2 3 4 6

(1, 6)

(2, 3)
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

WEEK 14

/*

QUESTION 14.1-

Given a number n, write an algorithm and a program to find nth ugly number. Ugly numbers are those numbers whose only prime factors are 2, 3 or 5. The sequence 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24,. is sequence of ugly numbers.

Input: First line of input will give number of test cases T. For each test case T, enter a number n.

Output: There will be T output lines. For each test case T, Output will be nth ugly number
*/

```
#include<bits/stdc++.h>

using namespace std;


int findUgly(int n) {

    int ugly[n];

    ugly[0] = 1;

    int i2 = 0, i3 = 0, i5 = 0;

    int next2 = 2;

    int next3 = 3;

    int next5 = 5;

    int next = 1;

    for (int i = 1; i < n; i++) {

        next = min(next2, min(next3, next5));

        ugly[i] = next;

        if (next == next2) {

            i2++;

            next2 = ugly[i2] * 2;

        }

        if (next == next3) {
            i3++;
            next3 = ugly[i3] * 3;
        }
        if (next == next5) {
            i5++;
            next5 = ugly[i5] * 5;
        }
    }
    return next;
}

int main() {
    int T;
    cin >> T;
    while (T--) {
        int n;
        cin >> n;
        cout << findUgly(n) << endl;
    }
    return 0;
}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

Input:

```
4

3

8

12

20
```

Output:

```
3

12

18

30
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
/*

  QUESTION 14.2-

  Given a directed graph, write an algorithm and a program to find mother vertex in a

   graph. A mother vertex is a vertex v such that there exists a path from v to all other

   vertices of the graph.

   Input:

   Graph in the form of adjacency matrix or adjacency list is provided as an input

  Output will be the mother
*/


  #include<bits/stdc++.h>

  using namespace std;


  class Graph {

      int V;

      list<int> *adj;

      void DFSUtil(int v, vector<bool> &visited);

  public:

      Graph(int V);

      void addEdge(int v, int w);

      int findMother();

  };


  Graph::Graph(int V) {

      this->V = V;

      adj = new list<int>[V];

  }


  void Graph::addEdge(int v, int w) {
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
        adj[v].push_back(w);
    }


    void Graph::DFSUtil(int v, vector<bool> &visited) {
        visited[v] = true;
        for(auto i = adj[v].begin(); i != adj[v].end(); ++i) {
            if(!visited[*i])
                DFSUtil(*i, visited);
        }
    }


    int Graph::findMother() {
        vector<bool> visited(V, false);
        int v = 0;
        for(int i = 0; i < V; i++) {
            if(!visited[i]) {
                DFSUtil(i, visited);
                v = i;
            }
        }
        fill(visited.begin(), visited.end(), false);
        DFSUtil(v, visited);
        for(int i = 0; i < V; i++) {
            if(!visited[i])
                return -1;
        }
        return v;
    }
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

```
int main() {

    int V;

    cin >> V;

    Graph g(V);

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

        int vertex, num;

        cin >> vertex;

        cin >> num;

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

            int adjVertex;

            cin >> adjVertex;

            g.addEdge(vertex, adjVertex);

        }

    }

    int mother = g.findMother();

    if(mother != -1)

        cout << mother << endl;

    else

        cout << "No mother vertex found." << endl;

    return 0;

}
```

NAME: SHARVI NEGI
SECTION: ML (53)
UNIVERSITY ROLL NUMBER: 2021911

OUTPUT

Input:

```
6

0 1 1

1 1 2

2 1 3

3 1 4

4 1 5

5 1 0
```

Output:

```
No mother vertex found.
```

Input:

```
5

0 1 1

1 1 2

2 1 3

3 1 4

4 1 0
```

Output:

```
0
```