Write a program to find two pairs of integer in the given array whose sum are equal, using hashing technique. If more than two pairs are found, then output the pairs which appears first in the array in the basis of it's index.

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

bool findPairs(int arr[], int n)
{
	map<int, pair<int, int> > Hash;
	for (int i = 0; i < n; ++i)
	{
		for (int j = i + 1; j < n; ++j)
		{
			int sum = arr[i] + arr[j];
			if (Hash.find(sum) == Hash.end())
				Hash[sum] = make_pair(i, j);

			else
			{
				pair<int, int> pp = Hash[sum];

				cout << arr[pp.first] << "+" << arr[pp.second]
					<< " and " << arr[i] << "+" << arr[j] ;
				return true;
			}
		}
	}

	cout << "No pairs found";
	return false;
}
```

```
// Driver program
int main()
{
    int n;
    cin >> n;
        int arr[n];
        for(int i = 0; i< n ;i++)
        {
            cin >> arr[i];
        }
        findPairs(arr, n);
        return 0;
}
```

Write a program to count distinct elements in an array within the window size of W, using Hashing technique

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

void countDistinct(int arr[], int k, int n)
{
        map<int, int> hm;

        int dist_count = 0;
        for (int i = 0; i < k; i++)
        {
        if (hm[arr[i]] == 0)
        {
                dist_count++;
        }
        hm[arr[i]] += 1;
```

```
        }
cout << dist_count << endl;
for (int i = k; i < n; i++)
{
        if (hm[arr[i-k]] == 1)
        {
                dist_count--;
        }
        hm[arr[i-k]] -= 1;

        if (hm[arr[i]] == 0)
        {
        dist_count++;
        }
        hm[arr[i]] += 1;

        cout << dist_count << endl;
}
}

int main()
{
   int n;
   cin >> n;
   int arr[n];
   for(int i=0;i<n;i++)
   {
      cin >> arr[i];
   }
   int k;
   cin >> k;
   countDistinct(arr, k, n);
```

```
    return 0;
}
```

Radha took the school children to trip, where they are instructed to be in pairs. Each student posses a chest number which can be repeated. Radha has to identify the two pair of students. Where the chest number of student in the first row of the pair 1 and the chest number of student in the second row of the pair 2 are same, similarly the chest number of student in the second row of the pair 1 and the chest number of student in the first row of the pair 2 are same. Radha is clear that it can be done using hashing techniques efficiently.

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

void findSymPairs(vector<vector<int> > arr, int row)
{
        map<int, int> hM;
        int s = 0;

        for (int i = 0; i < row; i++)
        {
                int first = arr[i][0];
                int sec = arr[i][1];

                if (hM.find(sec) != hM.end() && hM[sec] == first)
                {
                   cout << "(" << sec << ", " << first << ")" <<endl;
                        s = 1;
                }

                else
                        hM[first] = sec;
        }
```

```
        if(s==0)
        {
            cout<<"No such pairs"<<endl;
        }
}

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

        vector<vector<int> > arr( n , vector<int> (2, 0));
        for(i=0;i<n;i++)
        {
            for(int j = 0;j<2;j++)
            {
                cin >> arr[i][j];
            }
        }
        findSymPairs(arr, n);
}
```

Write a program to find the pairs of integers in the array which can be sum up to the given number S.

Note: Use hashing technique to solve this.

```
import java.io.*;
import java.util.HashSet;
import java.util.*;

class PairSum {
        static void printpairs(int arr[], int sum)
        {
                HashSet<Integer> s = new HashSet<Integer>();
```

```
            int status = 0;
            for (int i = 0; i < arr.length; ++i) {
                int temp = sum - arr[i];

                if (s.contains(temp)) {
                    System.out.println("Pairs: "+ arr[i] + " and " + temp);
                    status = 1;
                }
                s.add(arr[i]);
            }
            if(status == 0)
            {
               System.out.println("No such pairs");
            }
        }

        public static void main(String[] args)
        {
           Scanner sc = new Scanner(System.in);
           int n;
           n = sc.nextInt();
           int[] A = new int[n];
                for(int i = 0;i<n;i++)
                {
                   A[i] = sc.nextInt();
                }
                int m;
                m = sc.nextInt();
                printpairs(A, m);
        }
}
```

Write a program to find whether the array is a subset or not using Hashing technique.

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

void quickSort(int *arr, int si, int ei);
int binarySearch(int arr[], int low, int high, int x);

bool isSubset(int arr1[], int arr2[], int m, int n)
{
	int i = 0;

	quickSort(arr1, 0, m-1);
	for (i=0; i<n; i++)
	{
		if (binarySearch(arr1, 0, m - 1, arr2[i]) == -1)
		return 0;
	}

	return 1;
}

int binarySearch(int arr[], int low,
                                        int high, int x)
{
	if(high >= low)
	{
		int mid = (low + high)/2;
		if(( mid == 0 || x > arr[mid-1]) && (arr[mid] == x))
			return mid;
		else if(x > arr[mid])
			return binarySearch(arr, (mid + 1), high, x);
		else
			return binarySearch(arr, low, (mid -1), x);
```

```
        }
        return -1;
}

void exchange(int *a, int *b)
{
        int temp;
        temp = *a;
        *a = *b;
        *b = temp;
}

int partition(int A[], int si, int ei)
{
        int x = A[ei];
        int i = (si - 1);
        int j;

        for (j = si; j <= ei - 1; j++)
        {
                if(A[j] <= x)
                {
                        i++;
                        exchange(&A[i], &A[j]);
                }
        }
        exchange (&A[i + 1], &A[ei]);
        return (i + 1);
}

void quickSort(int A[], int si, int ei)
{
```

```
        int pi;
        if(si < ei)
        {
                pi = partition(A, si, ei);
                quickSort(A, si, pi - 1);
                quickSort(A, pi + 1, ei);
        }
}

int main()
{
   int m,n;
   cin >> m;
   cin >> n;
   int arr1[m];
   int arr2[n];
   for(int i = 0; i<m;i++)
   {
        cin >> arr1[i];
   }
   for(int i = 0; i<n;i++)
   {
      cin >> arr2[i];
   }

        if(isSubset(arr1, arr2, m, n))
                cout << "Subset";
        else
                cout << "Not a subset";
        return 0;
}
```