**ASSIGNMENT-3**

1. Given two arrays: a1[0..n-1] of size n and a2[0..m-1] of size m. Task is to check whether a2[] is a subset of a1[] or not. Both the arrays can be sorted or unsorted. It may be assumed that elements in both array are distinct.

Code:

// C++ program to check if an array is a subset of another array

#include<bits/stdc++.h>

using namespace std;

bool Subset(int arr1[], int arr2[],

int m, int n)

{

int i = 0;

int j = 0;

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

{

for (j = 0; j < m; j++)

{

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

break;

}

if (j == m)

return 0;

}

return 1;

}

int main()

{

int m,n;

cout << “\nEnter the size of array 1 : “;

cin >> m;

cout << “\nEnter the size of array 2 : “;

cin >> n;

int arr1[m],arr2[n];

int i;

cout << “\nEnter the array 1 elements : “;

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

{

cin >> arr1[i];

}

cout << “\nEnter the array 2 elements : “;

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

{

cin >> arr2[i];

}

if(Subset(arr1, arr2, m, n))

cout << “\nArray2 is a subset of Array1\n “;

else

cout << “\nArray2 is not a subset of Array1\n”;

getchar();

return 0;

}

2. Given an array arr of size n and an integer X. Find if there's a triplet in the array which sums up to the given integer X.

Code:

#include <bits/stdc++.h>

using namespace std;

bool Triplet(int M[], int n, int sum)

{

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

{

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

{

for (int k = j + 1; k < n; k++)

{

if (M[i] + M[j] + M[k] == sum)

{

cout << "The triplet is << " {" << M[i] <<

", " << M[j] << ", " << M[k]<<"}"<<"in the array sums upto"<<sum;

return 1;

}

}

}

}

return 0;

}

int main()

{

int M[] = { 1 ,4, 45, 6, 10, 8};

int sum = 13;

int n = sizeof(M) / sizeof(M[0]);

Triplet(M, n, sum);

return 0;

}

3. A conveyor belt has packages that must be shipped from one port to another within days.

Code:

#include <bits/stdc++.h>

using namespace std;

bool isTrue(int presentweight[], int n,int DAY, int mx)

{

int st = 1;

int sum = 0;

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

sum += presentweight[i];

if (sum > mx) {

st++;

sum = presentweight[i];

}

if (st > DAY) return false; }

return true;

}

void totalDAY(int presentweight[], int DAY,int n)

{

int sum = 0;

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

sum += presentweight[i];

int s = presentweight[0];

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

s = max(s, presentweight[i]);

}

int e = sum;

int res = -1;

while (s <= e) {

int miday = s + (e - s) / 2;

if (isTrue(presentweight, n, DAY, miday)) {

res = miday;

e = miday - 1;

}

else

s = miday + 1;

}

cout << res;

}

int main()

{

int presentweight[] = { 1,2,3,4,5,6,7,8,9,10 };

int DAY = 5;

int N = sizeof(presentweight) / sizeof(presentweight[0]);

totalDAY(presentweight, DAY, N);

return 0;

}