10) TUG OF WAR:

LINK OF EXPLANATION: Minimum Subset Sum Difference Explained | Tug Of War | Recur..

METHOD:

1) The question tells us that we have to find the two subsets of the given array such that the difference of the sum of the elements of the two subsets is minimum.
2) So here also, we have the choice/selection process like knapsack problem.
3) In this question we find all the possible combinations of the two subsets having equal elements and evaluate their diff of sum of their elements.
4) If the diff is less than the previously stored mindiff, then the answer is changed to this new subsets and the mindiff is also reassigned with the new minimum value.
5) So we pass the two subsets and their resultant sum every time recursively.
6) Also if one of the arrays contains n/2 elements then all the remaining elements will go into the other subset . so the choice of selection occurs in the case when both the subsets contain elements less than n/2.
7) The backtrack condition is the popping of the element which is pushed in one of the subset and pushing them into the other subset.

CODE OF THE PROGRAM:

```
#include<iostream>
#include<vector>
#include<limits.h>
using namespace std;

int minsumdiff=INT_MAX;
vector<int> answer1,answer2;

void fun(vector<int>&arr,int &n,int index,int sum1,int sum2,vector<int>&a,vector<int>&b){
  if(index==n){
    if(abs(sum1-sum2)<=minsumdiff){
      minsumdiff=abs(sum1-sum2);
      answer1=a;
      answer2=b;
    }
    else{
      return;
    }
  }
  else if(n%2==0){
    if(a.size()==n/2){
      b.push_back(arr[index]);
      fun(arr,n,index+1,sum1,sum2+arr[index],a,b);
      b.pop_back();
    }
    else if(b.size()==n/2){
      a.push_back(arr[index]);
      fun(arr,n,index+1,sum1+arr[index],sum2,a,b);
```

```
            a.pop_back();
        }
        else{
            a.push_back(arr[index]);
            fun(arr,n,index+1,sum1+arr[index],sum2,a,b);
            a.pop_back();
            b.push_back(arr[index]);
            fun(arr,n,index+1,sum1,sum2+arr[index],a,b);
            b.pop_back();
        }
    }
    else if(n%2==1){
        if(a.size()==n/2+1){
            b.push_back(arr[index]);
            fun(arr,n,index+1,sum1,sum2+arr[index],a,b);
            b.pop_back();
        }
        else if(b.size()==n/2+1){
            a.push_back(arr[index]);
            fun(arr,n,index+1,sum1+arr[index],sum2,a,b);
            a.pop_back();
        }
        else{
            a.push_back(arr[index]);
            fun(arr,n,index+1,sum1+arr[index],sum2,a,b);
            a.pop_back();
            b.push_back(arr[index]);
            fun(arr,n,index+1,sum1,sum2+arr[index],a,b);
            b.pop_back();
        }
    }
}

int main(){
    int n;
    cout<<"\n Enter the number of elements in the array:";
    cin>>n;
    vector<int> arr(n);
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }
    vector<int> a;
    vector<int> b;
    int sum1=0;
    int sum2=0;
    fun(arr,n,0,sum1,sum2,a,b);
    cout<<"\n Subset 1 : ";
    for(int i=0;i<answer1.size();i++){
        cout<<answer1[i]<<" ";
    }
```

```
    cout<<"\n Subset 2 : ";
    for(int i=0;i<answer2.size();i++){
        cout<<answer2[i]<<" ";
    }
    return 0;
}
```