VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

DSA Final Project Review

Fair Property Division

Team members -

Name	Reg. No.
Dayeem Parkar	19BCI0001
Deepak Pandey	19BCI0003
Akashdeep	19BCI0036

Problem statement:-

A father has passed away and in his will has left a number of properties to his multiple children. The only direction in the will is to divide the property as evenly as possible".

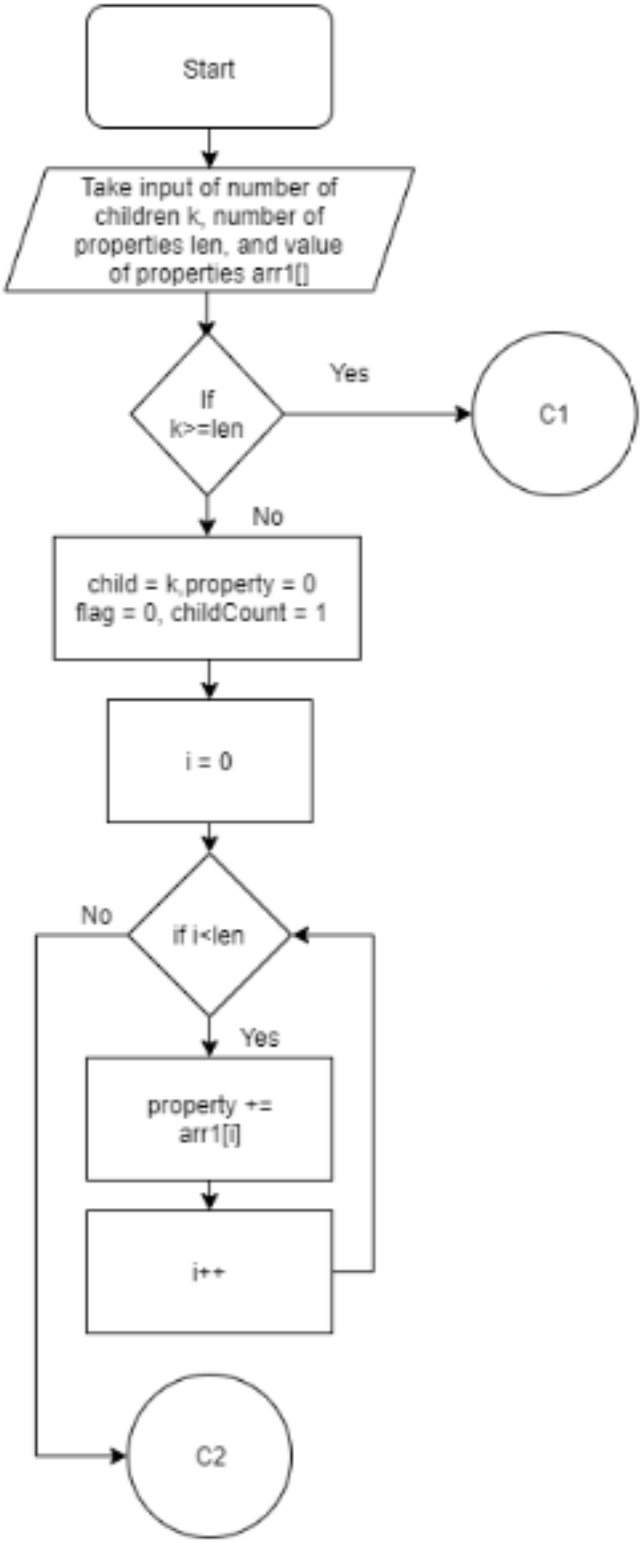
As the executor of the will, you have been given the value of each property. You are to decide how to divide up the property to the children to minimize the difference as much as possible.

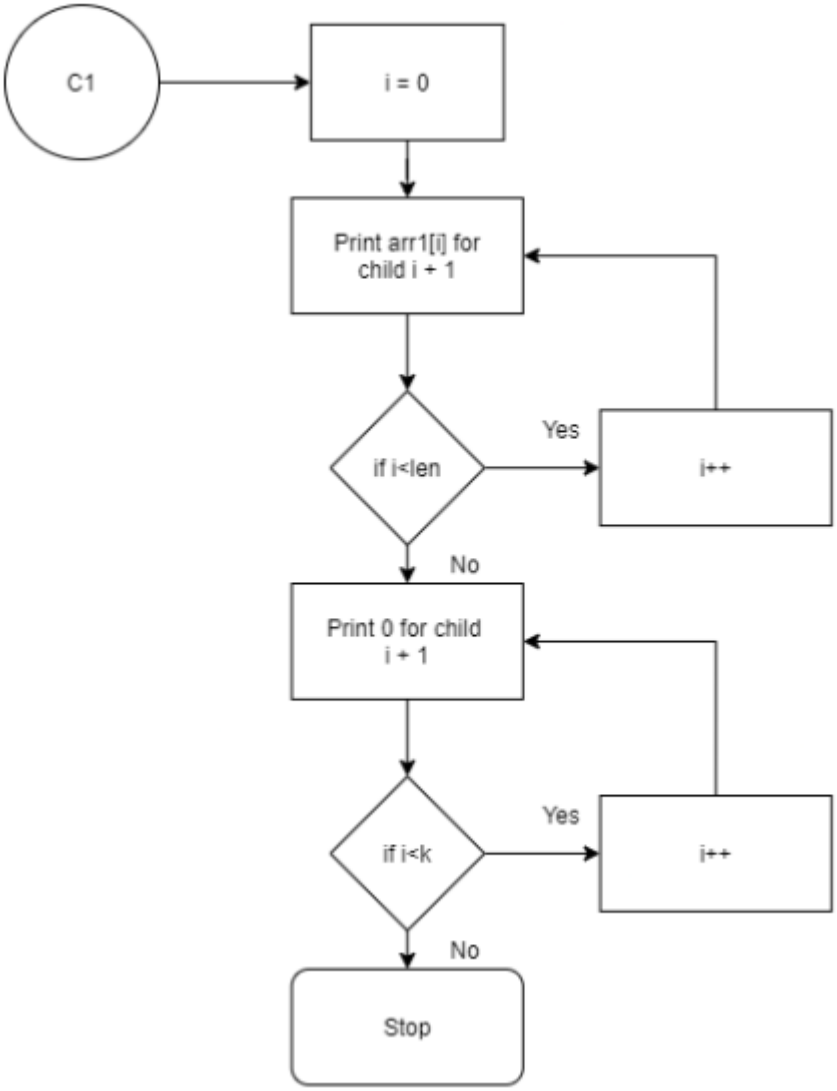
Solution using recursive method:-

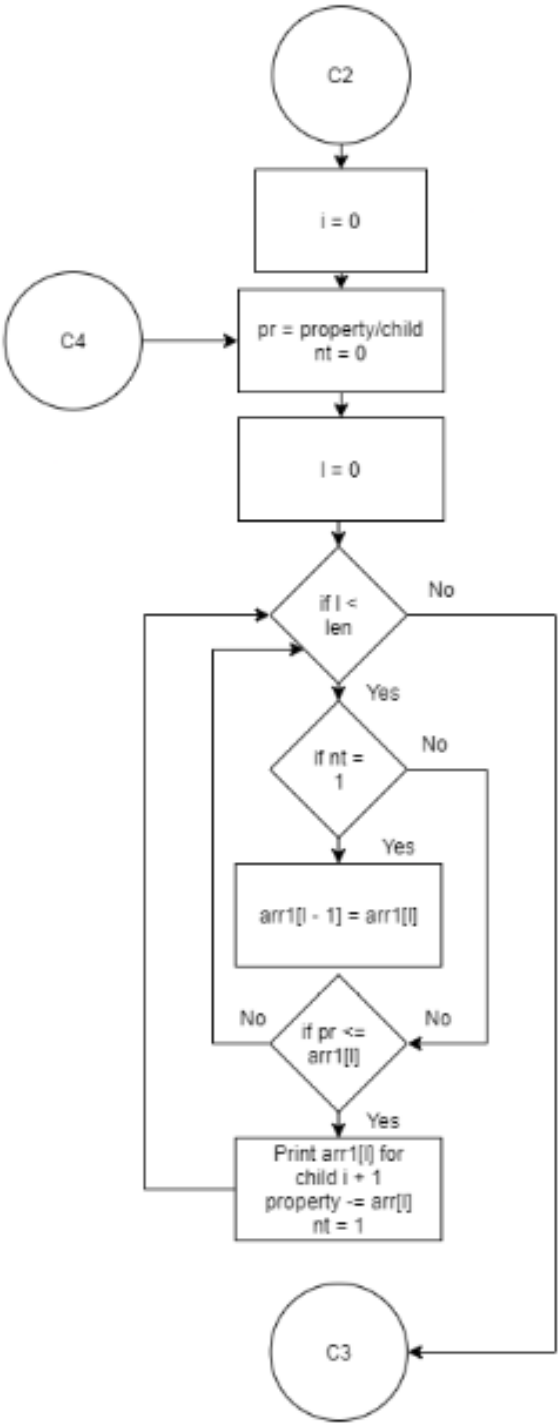
- The recursive approach is to generate all possible sums from all the values of the properties and to check which solution is the most optimal one.
- If there are n children, that many subsets have to be created and checked.
- So, we end up finding the minimum difference out of all the possible property divisions.

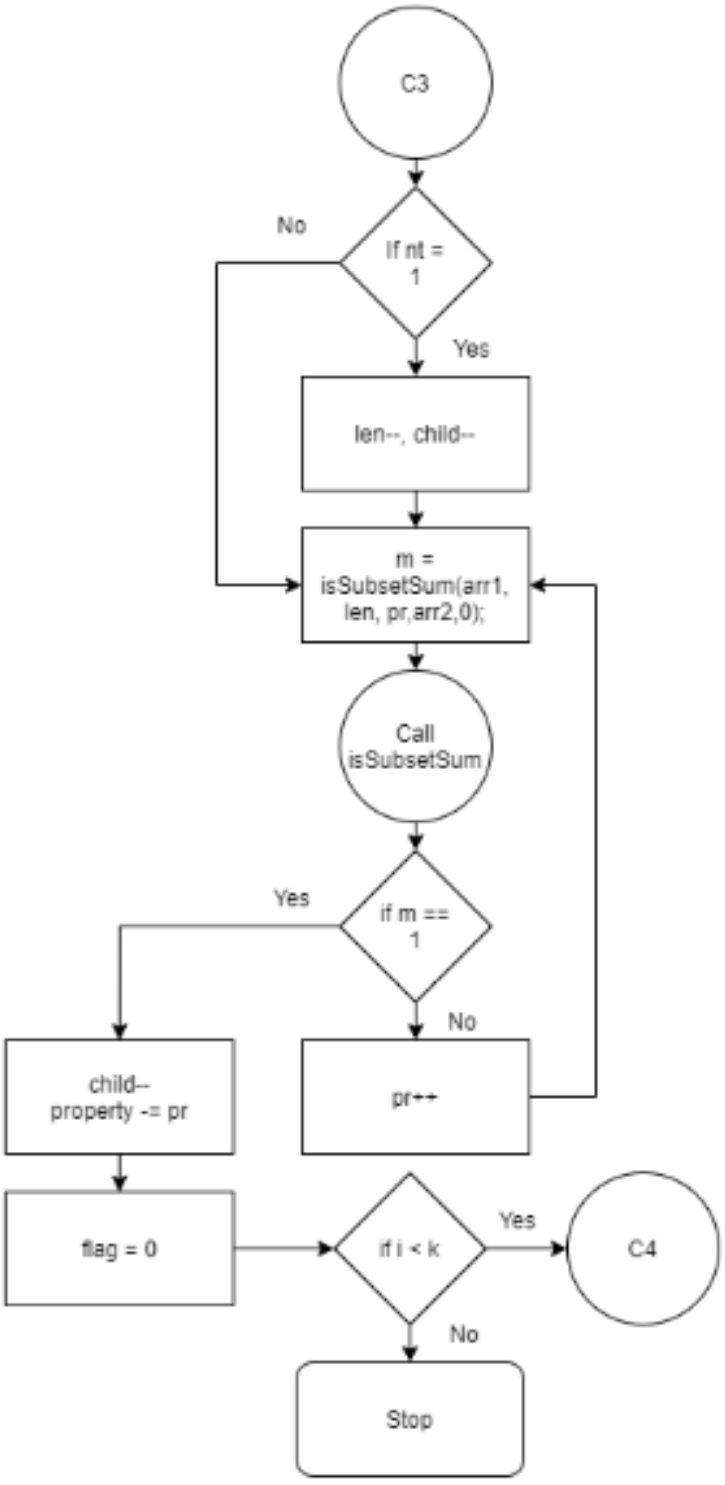
Flow Chart:-

Main function

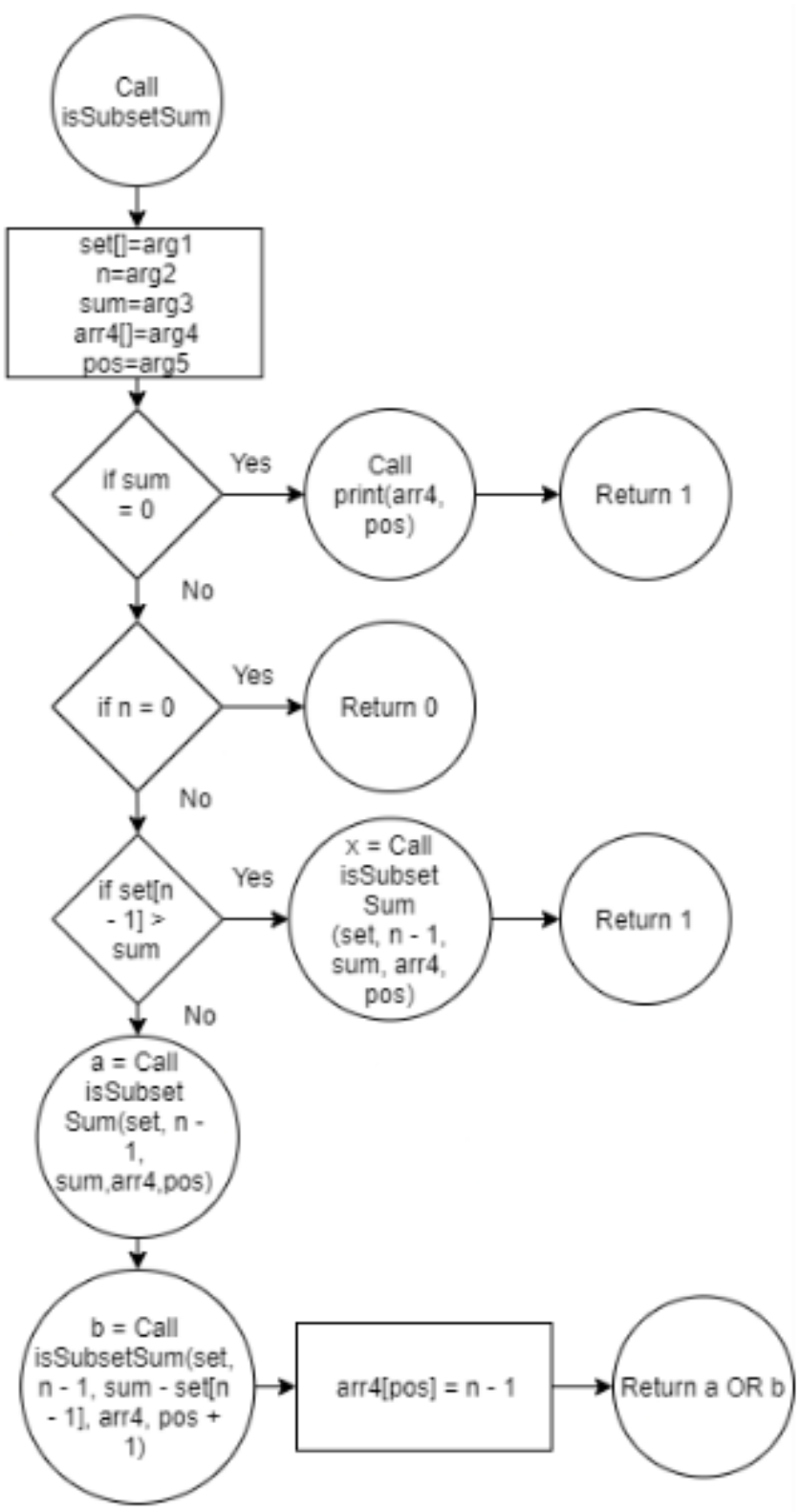



C1
i = 0
Print arr1[i] for child i + 1
if i<len
Yes
i++
No
Print 0 for child i + 1
if i<k
Yes
i++
No
Stop

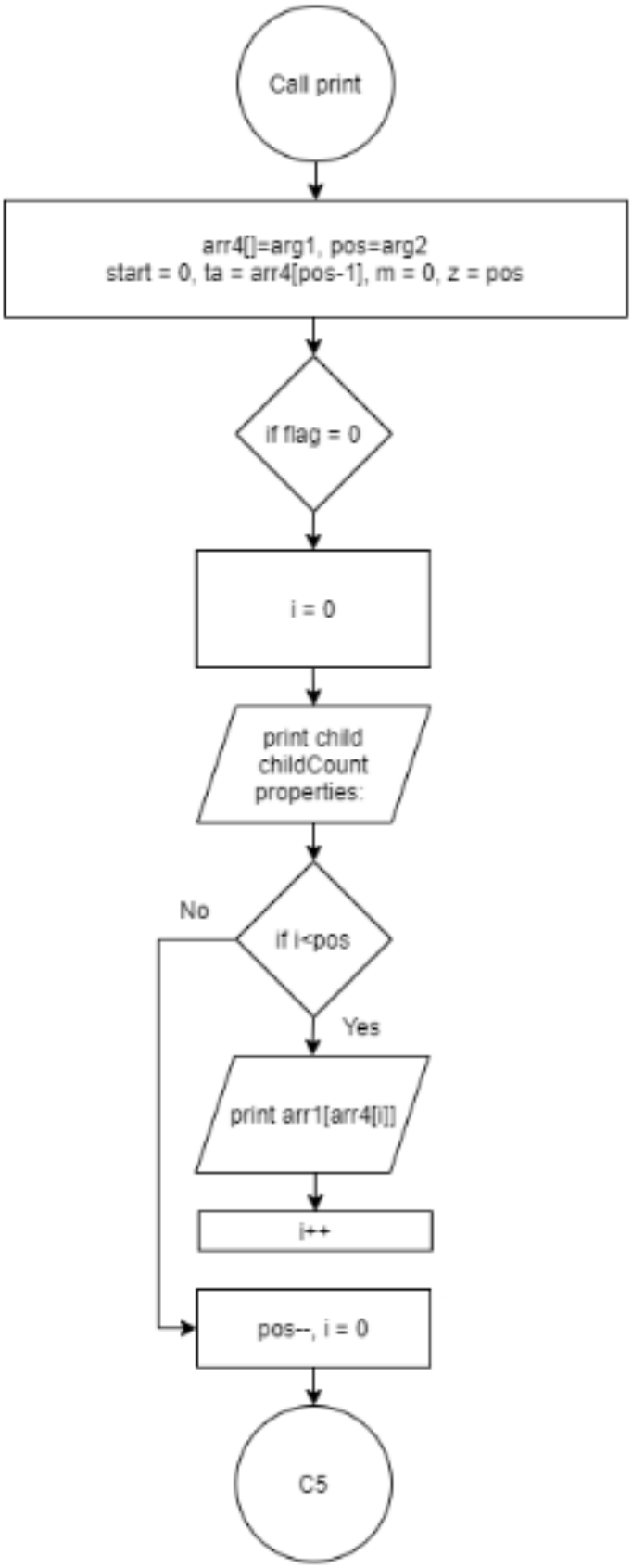

C2
i = 0
C4
pr = property/child
nt = 0
l = 0
if l <
len
No
Yes
if nt =
1
No
Yes
arr1[l - 1] = arr1[l]
No
if pr <=
arr1[l]
No
Yes
Print arr1[l] for
child i + 1
property -= arr[l]
nt = 1
C3

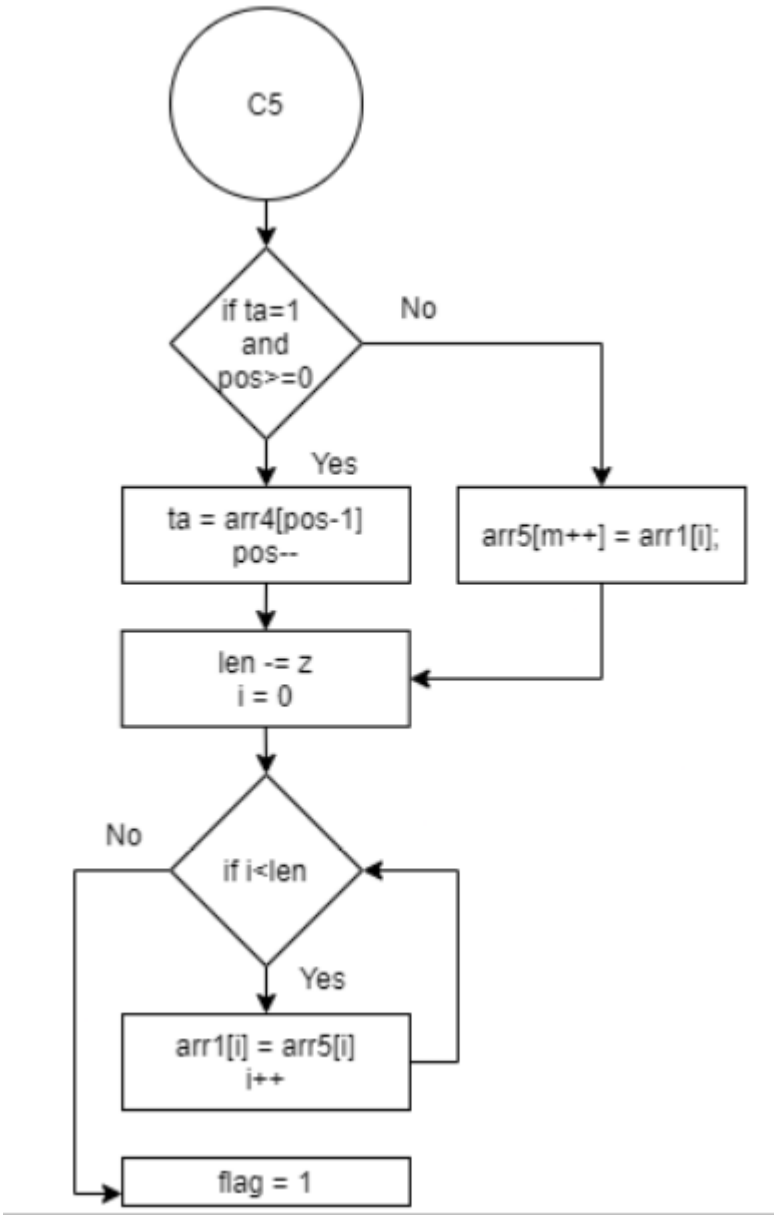

C3
If nt = 1
No
Yes
len--, child--
m = isSubsetSum(arr1, len, pr,arr2,0);
Call isSubsetSum
if m == 1
Yes
No
child-- property -= pr
pr++
flag = 0
if i < k
Yes
C4
No
Stop

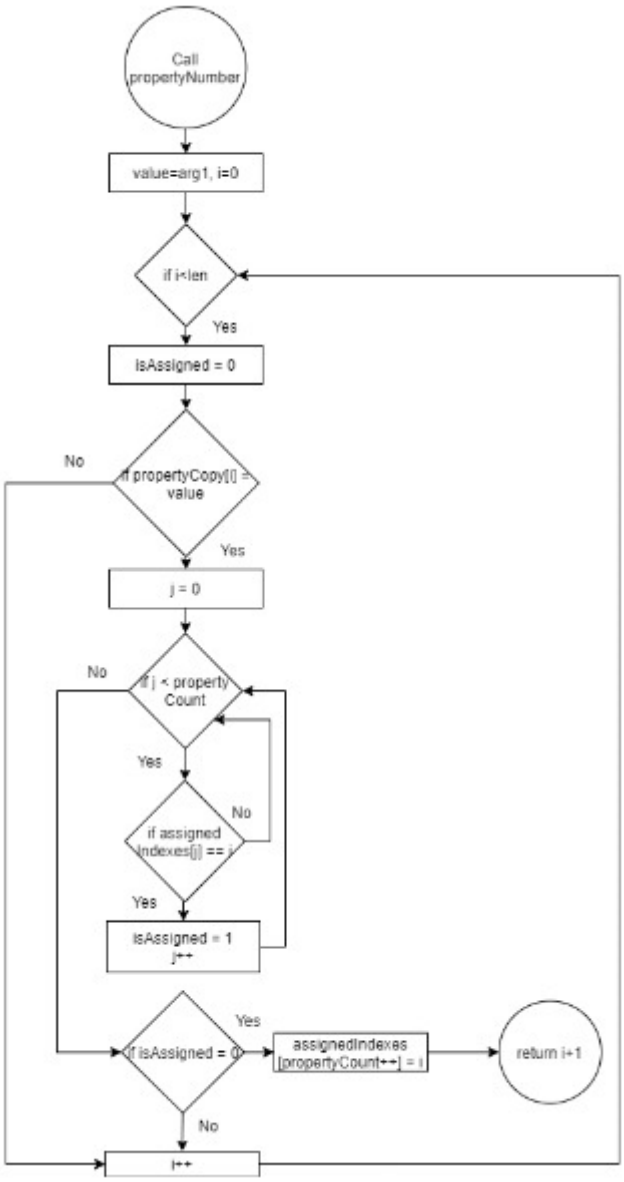
isSubsetSum function:-



Print function:-




C5
if ta=1
and
pos>=0
No
Yes
ta = arr4[pos-1]
pos--
arr5[m++] = arr1[i];
len -= z
i = 0
No
if i<len
Yes
arr1[i] = arr5[i]
i++
flag = 1


Call propertyNumber
value=arg1, i=0
if i<len
Yes
isAssigned = 0
No
if propertyCopy[i] = value
Yes
j = 0
No
if j < property Count
Yes
No
if assigned Indexes[j] == j
Yes
isAssigned = 1
j++
Yes
if isAssigned = 0
assignedIndexes [propertyCount++] = i
return i+1
No
i++

Algorithm:

1) Start
2) Take the inputs for no child, no of properties
3) Then input for each property.
4) If the no or properties is less than equal to no of child
    - Then give one property to one child
    - If any child is left than assign them 0 value.
5) Calculate sum of property array
6) Calculate average of property array based on child left to assign property.
7) If any property value is greater than current child then directly assign that value to current child.
8) Otherwise calculate the nearest property value to average which can be assigned to current child. That involve following steps
    - Call recursively until value of sum does not become zero and return 1
    - Or if the array size becomes zero in that case return 0.
    - If above conditions do not match then we check if the property value is greater than sum or not.
    - If it is greater then we leave that value and check for further values
    - Next we subtract the current array value from sum
    - Call recursively this function.
9) Then we subtract left child by 1.
10) Subtract property value by the holdings of current child.
11) Go to step 6
12) Finish

Code:–

```
#include <stdio.h>

#include<limits.h>

#include<stdlib.h>


// Global variable for child and property

int arr1[1000], arr2[1000], arr5[1000], propertyCopy[1000],
assignedIndexes[1000], len, lenCopy, flag = 0, childCount = 1, propertyCount =
0;


//Fetches property numer when its value is provided

int propertyNumber(int value){

        int i, j, isAssigned;

        for(i = 0; i < lenCopy; i++){

                isAssigned = 0;

                if(propertyCopy[i] == value){  //if value matches

                        for(j = 0; j < propertyCount; j++){

                                if(assignedIndexes[j] == i)  //check if it is already
assigned(for >1 properties of same value

                                        isAssigned = 1;

                        }

                        if(isAssigned == 0){  //if it is not assigned, add to assigned
array

                                assignedIndexes[propertyCount++] = i;
```

```
                    return i + 1;
                }
            }
        }
        return 0;  //safety net
}

void print(int arr4[], int pos){
   int start = 0, i, ta = arr4[pos-1], m = 0, z = pos,sum1=0;
        if(flag == 0){
      // By this the user get intimated about the child holding
      printf("\nProperties given to child %d:", childCount++);
      for(i = 0; i < pos; i++){
         printf("\nProperty: %d, Value: %d", propertyNumber(arr1[arr4[i]]),
arr1[arr4[i]]);
         sum1+=arr1[arr4[i]];
      }
      // pos variable keep track of no of properties child have
      pos--;
      printf("\nTotal Value = %d\n",sum1);
      // then we update property array with the left out properties
      for(i = 0; i < len; i++){
         if(ta == i && pos >= 0){
            ta = arr4[pos-1];
```

```
            pos--;
          }
                        else
            arr5[m++] = arr1[i];
       }
       // Also we update the left out length of property array
       len = len - z;
       for(i = 0; i < len; i++)
          arr1[i] = arr5[i];
       // flag is updated to 1 so that no child get properties two times
       flag = 1;
    }
}

// This function will decide if we can give the required sum exactly from the
array
int isSubsetSum(int set[], int n, int sum, int arr4[],int pos){
   int a, b;

         // Base Cases
   if(sum == 0){
      print(arr4, pos);
      return 1;
   }
```

```
   // return 0 if all array is traversed
   if(n == 0)
      return 0;
   // if current value is greater than sum then we leave that and process fro
next element
   if(set[n - 1] > sum)
      return isSubsetSum(set, n - 1, sum, arr4, pos);
   a = isSubsetSum(set, n - 1, sum,arr4,pos);
   // If not
   // We add to the property array of child
   arr4[pos] = n - 1;
   b = isSubsetSum(set, n - 1, sum - set[n - 1], arr4, pos + 1);
   // return the true if required sum can be achieved
   return a || b;
}



int main(){
        int i, l, property = 0, child, pr, nt, m, k;
        printf("CSE2003 Data Structures and Algorithms Project: Fair Property
Division\nMembers:");
        printf("\n19BCI0001: Dayeem Parkar\n19BCI0003: Deepak
Pandey\n19BCI0036: Akashdeep");
        printf("\n\nEnter the number of children: ");
```

```
        scanf("%d", &k);
        printf("\nEnter the number of properties: ");
        scanf("%d", &len);
        lenCopy = len;
        for(i = 0; i < len; i++){
                printf("Enter value of property %d: ", i + 1);
                scanf("%d", &arr1[i]);
                propertyCopy[i] = arr1[i];
        }
   // no of child > length of array
   // then give each value to different child
   if(k>=len){
                for(i = 0; i < len; i++){
         printf("\nProperty given to child %d:\nProperty: %d, Value: %d\n", i + 1,
propertyNumber(arr1[i]), arr1[i]);
         printf("Total Value = %d\n",arr1[l]);
       }
       for(i = len; i < k; i++){
         printf("\nNo property given to child %d:\nValue: 0\n", i + 1);
         printf("Total Value = 0\n");
       }
         }
         else{
            child = k;
```

```
    // get the sum of property array
    for(i = 0; i < len; i++)
property += arr1[i];
    for(i = 0; i < k; i++){
// get the average of property
pr = property/child;
nt = 0;
for(l = 0; l < len; l++){
    // first process if any of property value is greater than average
    // if that's the case give to current child
    if(nt == 1)
        arr1[l-1]=arr1[l];
    else if(pr <= arr1[l]){
        printf("\nProperty given to child %d:\nProperty: %d, Value: %d\n", i
+ 1, propertyNumber(arr1[i]), arr1[l]);
        printf("Total Value = %d\n",arr1[l]);
        childCount++;
        property = property - arr1[l];
        nt = 1;
    }
}
if(nt == 1){
    len--;
    child--;
```

```
            continue;
        }
        // If that's not the case then process for suitable value which can be
given to child
        while(1){
            int m = isSubsetSum(arr1, len, pr,arr2,0);
            if(m == 1){

                child--;
                property = property-pr;
                break;
            }
            pr++;
        }
        flag = 0;
          }
        }
        return 0;
}
```

Output:-

```
CSE2003 Data Structures and Algorithms Project: Fair Property Division
Members:
19BCI0001: Dayeem Parkar
19BCI0003: Deepak Pandey
19BCI0036: Akashdeep

Enter the number of children: 4

Enter the number of properties: 6
Enter value of property 1: 120
Enter value of property 2: 80
Enter value of property 3: 180
Enter value of property 4: 40
Enter value of property 5: 90
Enter value of property 6: 130

Property given to child 1:
Property: 3, Value: 180
Total Value = 180

Properties given to child 2:
Property: 4, Value: 40
Property: 1, Value: 120
Total Value = 160

Properties given to child 3:
Property: 5, Value: 90
Property: 2, Value: 80
Total Value = 170

Property given to child 4:
Property: 6, Value: 130
Total Value = 130

Process returned 0 (0x0)   execution time : 44.211 s
Press any key to continue.
```