```
//N queen
#include <bits/stdc++.h>
#define n 4
using namespace std;

int a[n + 1][n + 1];
int totalSolutions = 0;

bool is_safe(int row, int col) {
   // Check for queens in the same column
   for (int i = 1; i <= row; i++) {
       if (a[i][col] == 1) return false;
   }

   // Check for queens in the left upper diagonal
   for (int i = row, j = col; i >= 1 && j >= 1; i--, j--) {
       if (a[i][j] == 1) return false;
   }

   // Check for queens in the right upper diagonal
   for (int i = row, j = col; i >= 1 && j <= n; i--, j++) {
       if (a[i][j] == 1)  return false;
   }
   return true;
}

void n_queen(int row) {
   if (row == n + 1) {
       for (int i = 1; i <= n; i++) {
           for (int j = 1; j <= n; j++) {
               cout << a[i][j] << " ";
           }cout << endl;
       }cout << endl;
       totalSolutions++;
   }

   for (int col = 1; col <= n; col++) {
       if (is_safe(row, col)) {
           a[row][col] = 1;
           n_queen(row + 1);
           a[row][col] = 0;
       }
   }
}

int main() {
   for (int i = 0; i <= n; i++) {
       for (int j = 0; j <= n; j++) {
           a[i][j] = 0;
       }
   }
   n_queen(1);
   cout << "Total solutions found: " << totalSolutions << endl;
}
```

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

void topo_sort(int vertices,
int edges) {
   vector<char> ans;
   queue<char> q;
   map<char, vector<char>>
graph;
   map<char, int> inDegree;

   vector<pair<char, char>>
edgeList = {
       {'A', 'B'},
       {'A', 'C'},
       {'B', 'D'},
       {'B', 'E'},
       {'C', 'E'},
       {'D', 'F'},
       {'E', 'F'}
   };

   for (int i = 0; i < edges; i++)
{
       char a = edgeList[i].first;
       char b =
edgeList[i].second;
       graph[a].push_back(b);
       inDegree[b]++;
   }

   for (char c = 'A'; c <= 'F';
c++) {
       if (inDegree[c] == 0) {
           q.push(c);
       }
   }

   while (!q.empty()) {
       char v = q.front();
       q.pop();
       ans.push_back(v);

       for (int i = 0; i <
graph[v].size(); i++) {
           char u = graph[v][i];
           inDegree[u]--;
           if (inDegree[u] == 0) {
               q.push(u);
           }
       }
   }

   for (int i = 0; i < ans.size();
i++) {
       cout << ans[i];
       if (i < ans.size() - 1) {
           cout << " ";
       }
   }
}

int main() {
   int vertices = 6;
   int edges = 7;

   topo_sort(vertices, edges);

   return 0;
}
```

```
//sum of subsets baba

#include <bits/stdc++.h>

using namespace std;


int tab[2000][2000];


int subsetSum(int a[], int n,
int sum) {

    if (sum == 0)

        return 1;


    if (n <= 0)

        return 0;


    if (tab[n - 1][sum] != -1)

        return tab[n - 1][sum];


    if (a[n - 1] > sum)

        return tab[n - 1][sum] =
subsetSum(a, n - 1, sum);

    else {

        return tab[n - 1][sum] =
subsetSum(a, n - 1, sum) ||
subsetSum(a, n - 1, sum - a[n
- 1]);

    }

}


void
totalSumAndListSubsets(int
set[], int n, int sum) {

    memset(tab, -1,
sizeof(tab)); // Fixed the
error by adding a closing
parenthesis


    if (subsetSum(set, n, sum))
{

        cout << "YES" << endl;

    }

    else {

        cout << "NO" << endl;

    }


    int totalSum = 0;

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

        totalSum += set[i];

    }

    cout << "Total sum of all
subsets: " << totalSum <<
endl;


    int totalSubsets = pow(2,
n);

    cout << "All possible
subsets: " << endl;

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

        cout << "{ ";

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

            if (i & (1 << j)) {

                cout << set[j] << " ";

            }

        }

        cout << "}" << endl;

    }

}


int main() {

    int n = 5;

    int a[] = {1, 5, 3, 7, 4};

    int sum = 12;


    totalSumAndListSubsets(a,
n, sum);


    return 0;

}
```

//sum of subsets tamal

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

const int N = 100005;
int arr[N], target;

int flag;
void f(int pos,
vector<int> &v, int
sum){
    if(sum == target){
        if(flag)
cout<<", ";
        else flag = 1;
        cout<<"{ ";
        for(int
i=0;i<v.size();i++) {
            cout<<v[i]
;
            if(i ==
v.size() - 1) cout<<"
";
            else
cout<<", ";
        }
        cout<<"}";
        return;
    }
    if(pos == -1)
return;
    f(pos - 1, v,
sum);
    v.push_back(arr[po
s]);
    f(pos - 1, v, sum
+ arr[pos]);
    v.pop_back();
}

int main(){
    int n;
    cin>>n>>target;
    for(int
i=0;i<n;i++){
        cin>>arr[i];
    }
    vector<int> v;
    f(n-1, v, 0);
}
/*
3 2
1 2 1
{ 2 }, { 1, 1 }
Process returned 0
(0x0)   execution time
: 4.181 s
Press any key to
continue.
*/
```