Assignment Solutions | Recursion - 2 | Week 11

1. Print all the elements of an array in reverse order.

Solution:

```
#include <bits/stdc++.h>
using namespace std;
void printElementsInReverse(vector<int> &arr, int currIndex, int n) {
    if(currIndex == n) {
        return;
    }
    printElementsInReverse(arr, currIndex+1, n);
    cout << arr[currIndex] << " ";
}

int main() {

    int n;
    cin >> n;

    vector<int> arr(n);
    for(int i = 0; i < n; ++i) {
        cin >> arr[i];
    }

    printElementsInReverse(arr, 0, n);

    return 0;
```

2. Print index of a given element in an array. If not present, print -1.

Solution:

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

int indexOfKey(vector<int> &arr, int currIndex, int n, int key) {
    if(currIndex == n) {
        return -1;
    }
    if(arr[currIndex] == key) {
        return currIndex;
    }
    return indexOfKey(arr, currIndex+1, n, key);
}

int main() {

    int n, key;
    cin >> n;

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

3. A function countAndSay is defined as:

   countAndSay(1) = “1”

   countAndSay(n) is the way you would "say" the digit string from countAndSay(n-1), which is then converted into a different digit string.

   So, if sample input is n = 4,

   countAndSay(1) = 1

   countAndSay(2) = “one 1” => 11

   countAndSay(3) = “two 1” => 21

   countAndSay(4) = “one 2 one 1” => 1211

Solution:

```
#include <bits/stdc++.h>
using namespace std;
string countAndSay(int n) {
    if(n == 1) {
        return "1";
    }
    string ans = "";
    string smallAns = countAndSay(n-1);
    for(int i = 0; i < smallAns.size();) {
        int count = 1;
        int j = i+1;
        while(j < smallAns.size() && smallAns[i] == smallAns[j])
   {
            j++;
            count++;
        }
        ans = ans + to_string(count) + smallAns[i];
        i = j;
    }
    return ans;
}

int main() {
```

4. Given an array of integers, print a sum triangle using recursion from it such that the first level has all array elements. After that, at each level the number of elements is one less than the previous

   level and elements at the level will be the sum of consecutive two elements in the previous level.

   So, if sample input is [5, 4, 3, 2, 1], sample output will be:

   [5, 4, 3, 2, 1]

   [9, 7, 5, 3]

   [16, 12, 8]

   [28, 20]

   [48]

Solution:

```cpp
#include <bits/stdc++.h>
using namespace std;
void sumTriangle(vector<int> &arr, int n) {
    if(n == 0) return;
    vector<int> temp(n-1);
    for(int i = 0; i < n; ++i) {
        cout << arr[i] << " ";
        if(i != 0) {
            temp[i-1] = arr[i-1] + arr[i];
        }
    }
    cout << endl;
    sumTriangle(temp, n-1);
}

int main() {

    int n;
    cin >> n;

    vector<int> arr(n);

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