Assignment Solutions | Recursion - 3 | Week 11

1. Write a recursive function to reverse a number. Avoid preceding 0s in the reversed number.

Solution:

```
#include <bits/stdc++.h>
using namespace std;
void reverseNum(int n, int &ans) {
    if(n == 0) {
        return;
    }
    int digit = n % 10;
    ans = ans * 10 + digit;
    reverseNum(n / 10, ans);
}

int main() {

    int n;
    cin >> n;

    int ans = 0;
    reverseNum(n, ans);
    cout << ans << endl;

    return 0;
}
```

2. Print all the increasing sequences of length k from first n natural numbers.

Solution:

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

void printIncreasingSequences(int curr, int n, int k, vector<int> &ans) {
    if(ans.size() == k) {
        for(int i = 0; i < k; ++i) {
            cout << ans[i] << " ";
        }
        cout << endl;
        return;
    }
    for(int i = curr+1; i <= n; ++i) {
        ans.push_back(i);
        printIncreasingSequences(i, n, k, ans);
        ans.pop_back();
    }
}

int main() {

    int n, k;
    cin >> n >> k;
```

3. Given two sorted arrays A and B, generate all possible arrays such that the first element is taken from A then from B then from A, and so on in increasing order till the arrays are exhausted. The generated arrays should end with an element from B.

   `A = {10, 15, 25}`

   `B = {1, 5, 20, 30}`

   `Output: {10 20}, {10 20 25 30}, {10 30}, {15 20}, {15 20 25 30}, {15 30}, {25 30}`

Solution:

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

void generateAlternateArrays(vector<int> &arr1, vector<int> &arr2, int i, int j,
vector<int> &temp, bool isFromA) {
    if(isFromA) {
        for(int k = 0; k < temp.size(); ++k) {
            cout << temp[k] << " ";
        }
        cout << endl;
    }

    if(isFromA) {
        for(int k = i; k < arr1.size(); ++k) {
            if(temp.size() == 0) {
                temp.push_back(arr1[k]);
                generateAlternateArrays(arr1, arr2, k+1, j, temp, !isFromA);
                temp.pop_back();
            } else {
                if(arr1[k] > temp.back()) {
                    temp.push_back(arr1[k]);
                    generateAlternateArrays(arr1, arr2, k+1, j, temp, !isFromA);
                    temp.pop_back();
```