16.2 Recursion Challenges

Q Check if an array is sorted or not (strictly Increasing)

{1, 2, 3, 4, 5, 6, 7}

array[0] < array[1] AND array(1...n) is sorted

Code:

```
bool sorted(int arr[], int n){

    if (n==1){
        return true;
    }
    bool restArray = sorted(arr+1, n-1);
    return (arr[0] < arr[1] && restArray);
}
```

Q Print numbers till n

1. Decreasing order — 9 8 7 6 5 4 3
2. Increasing order — 1 2 3 4 5 6 7

Code:

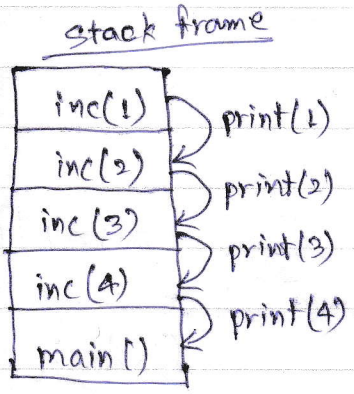
```
void dec(int n){
    if (n==0){      // base case
        return;
    }
    cout << n << endl;
    dec(n-1);
```

Stack frame

dec(1) → print(1)
dec(2) → print(2)
dec(3) → print(3)
dec(4) → print(4)
main()

```
void inc (int n) {

    if (n == 1) {                    // base case
        cout << "1" << endl;
        return;
    }
    inc (n-1);
    cout << n << endl;
}
```



Q Find the first and last occurence of a number in an array

{4, 2, 1, 2, 5, 2, 7}
I S
F L

2 → 1, 5

Code:

```
int firstocc (int arr [], int n, int i, int key) {
    if (i == n) {
        return -1;
    }
    if (arr [i] == key) {
        return i;
    }
    return firstocc (arr, n, i+1, key)
}
```

usage: firstocc (arr, arr-size, 0, key);

```
int lastocc (int arr[], int n, int i, int key){
    if (i==n){
        return -1;
    }
    int restArray = lastocc (arr, n, i+1, key);
    if (restArray != -1){
        return restArray;
    }
    if (arr[i] == key){
        return i;
    }
    return -1;
}
```

usage: `lastocc (arr, size_arr, 0, key);`