Algorithm Complexity

Problem 1: What is the time complexity of the function?

```
int sum(int n)
{
    int total = 0;
    for (int i = 0; i<=n; i++) total += i;
    return total;
}
```

Problem 2: What is the time complexity of the function?

```
int sum2(int n)
{
    int total = n*(n+1)/2;
    return total;
}
```

Problem 3: What is the time complexity of the function?

```
void fun(){
    int m = 100; int n = 1000;
    for (int i = 0; i<= m; i++){
        for (int j = 0; j <= n; j++) cout<< "Hello"<<endl;
    }
}
```

Problem 4: What is the time complexity of the function?

```
void all_pairs(int arr[], int size)
{
    for (int i = 0; i < size; i++)
    {
        for (int j = i; j< size; j++){
            if (i != j) cout << i << " "<<j<<endl;
        }
    }
}
```

Problem 5: What is the time complexity of the function?

```
int outputQ(int n)
{
    int counter = 0;
    for(int i=0 ; i < n ; i++ )
        for(int j=0 ; j < n ; j++ ) {
            counter++;
            cout << "Q";
            break;
        }
    return counter;
}
```

Problem 6: What is the time complexity of the function?

```
int countX(int n)
{
    int counter = 0;
    for(int i=1 ; i <= n ; i*=2 )
        for(int j=1; j <= n ; j++)
            for(int k=1; k <= n ; k++)
                counter++;
    return counter;
}
```

Problem 7: What is the time complexity of the function?

```
int binary_search(int arr[], int start, int end, int value)
{
    if(start > end) return -1;
    int middle = start + ((end - start) / 2);
    if (arr[middle] == value) return middle;
    if(arr[middle] > value)
        return binary_search(arr, start, middle - 1, value);
    return binary_search(arr, middle + 1, end, value);
}
```

Recursion

Problem 1: How many ways are there to choose k out of n things? (k<=n)

```
int choose(int k, int n)
{




}
```

Problem 2: Tower of Hanoi

```
//Only one disk can be moved at a time.
//Each move consists of taking the upper disk from one of the stacks
and placing it on top of another stack i.e. a disk can only be moved
if it is the uppermost disk on a stack.
//No disk may be placed on top of a smaller disk.

void solveHanoi(int n, int src, int dest, int buf)
{







}
```

Stack and Queue

Problem 1: Design a stack which can also return the minimum element in the stack.
Hint: You might want to use 2 stacks.

Problem 2: Implement a queue using two stacks.