

1 Introduction

Each exercise should be its own separate project.

Remember to follow the coding and documentation standards for the class listed on the MyClasses pages.

When you are ready to submit your work create a folder called Homework13 in that folder have separate folders for each project, one folder per project. Put all the code files needed for that project in its respective folder. Do not include the files that the IDE creates, I just want the code files. Zip the entire Homework13 folder up into a single zip file and submit it.

2 Exercises

1. You may recall that Pascal's triangle from your previous mathematics classes. The entries in Pascal's triangle are called combinations since if we want to know how many ways to select k objects from a set of n objects we simply go to the n^{th} row and k^{th} column of the triangle and read off the number. The row and column numbers start at 0. So to find out how many ways you can select 2 objects from a set of 5 we go to row number 5 and column number 2 and see that there are 10 ways to choose 2 from 5. We denote n choose k mathematically as

$$\binom{n}{k}$$

1										
1	1									
1	2	1								
1	3	3	1							
1	4	6	4	1						
1	5	10	10	5	1					
1	6	15	20	15	6	1				
1	7	21	35	35	21	7	1			
1	8	28	56	70	56	28	8	1		
1	9	36	84	126	126	84	36	9	1	
1	10	45	120	210	252	210	120	45	10	1

You may also recall that the way to get any entry in the middle of the triangle you simply take the sum of the entry above the one you are calculating and the entry one above and one to the left of the one you are calculating. So in mathematical terms,

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

Of course, if n or k is 0 then the value is 1, if $n = k$ the value is 1, and we will define anything outside these ranges the value is 0. That is, if n or k are negative or if $k > n$ we will simply have the program return 0. Create a function,

```
long comb(long n, long k)
```

That will return n choose k recursively by first checking the stopping conditions described above and then recursing on $n - 1$ choose k and $n - 1$ choose $k - 1$.

Then write a program that uses this recursive function to calculate n choose k . A couple runs are below.

```
Enter n and k (integer) with a space between them: 5 2
5 choose 2 = 10
```

```
Enter n and k (integer) with a space between them: 10 4
10 choose 4 = 210
```

```
Enter n and k (integer) with a space between them: 6 3
6 choose 3 = 20
```

```
Enter n and k (integer) with a space between them: 12 12
12 choose 12 = 1
```

```
Enter n and k (integer) with a space between them: 15 16
15 choose 16 = 0
```

```
Enter n and k (integer) with a space between them: 15 7
15 choose 7 = 6435
```

2. As you know, a Palindrome is a string that is the same written in either direction. For example, "A Toyota" or "Eva, can I see bees in a cave?". We can determine if a string is a palindrome recursively. Take a string and compare the first letter with the last letter, if the letters are the same you compare the second and second to last, and so on until you get to the middle of the string. Create a function,

```
bool isPal(const string& str, int startIndex, int endIndex)
```

that will return false if the characters in the start and end are not the same, true if it makes it to the middle of the string and recurses to the next letter positions to check otherwise. Create a program to check input palindromes using this recursive function. A couple runs are below.

```
Enter a string, no spaces and all lower case: evacaniseebeesinacave
evacaniseebeesinacave is a palindrome.
```

```
Enter a string, no spaces and all lower case: atoyota
atoyota is a palindrome.
```

```
Enter a string, no spaces and all lower case: help
help is not a palindrome.
```

3. In the study of chaos and dynamics there are two sequences, like the Fibonacci sequence but more complex, that come up. These are sometimes called Q numbers and D numbers. The D sequence is defined to be $D(1) = 1$, $D(2) = 1$, and

$$D(n) = D(D(n - 1)) + D(n - 1 - D(n - 2))$$

The Q sequence is defined to be $Q(1) = 1$, $Q(2) = 1$, and

$$Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2))$$

Obviously both of these functions are recursive.

Write a recursive function D that takes in a single long parameter n and returns a long that is the value of $D(n)$. Then write a recursive function Q that takes in a single long parameter n and returns a long that is the value of $Q(n)$. Once these functions have been written create a main that will ask the user for a value of n and print out each of the values of the Q and D functions from 1 to n . That is, $D(1), D(2), D(3), \dots, D(n)$ and the sequence $Q(1), Q(2), Q(3), \dots, Q(n)$. All of the printing is to be done in the main and no printing is to be done in the functions. A sample run is below,

```
Input n: 25
Q Numbers: 1 1 2 3 3 4 5 5 6 6 6 8 8 8 10 9 10 11 11 12 12 12 12 16 14
D Numbers: 1 1 2 2 2 3 4 4 4 4 5 6 7 8 8 8 8 8 9 10 10 10 11 13
```

4. In class we discussed the Towers of Hanoi puzzle and its recursive solution. We will extend this puzzle to one using 4 pegs instead of three. As you may suspect the 4 peg version can be solved in fewer moves since you will have two temporary pegs instead of just one.

Before we get into the 4 peg version take the code for the three peg version and add in the ability for the hanoi function to count the number of moves that have been made. This is probably easiest done by adding an integer parameter to the function, passed by reference, and each time a move is made increment the counter.

Now create another recursive function called hanoi4 that will do the 4 peg version. Here is how the 4 peg version works. If the number of disks is 0 then there is nothing to do, just return from the function. If the number of disks is 1, just move from the starting peg to the ending peg. If the number of disks is 2 or more then first solve the problem for $n - 2$ disks (yes, $n - 2$ not $n - 1$) from the start peg to the first temporary peg, using the second temporary peg and the end peg as the first and second temporary pegs respectively. Then do three moves, start peg to second temporary peg, start peg to end peg, then the second temporary peg to the end peg. Then solve the problem again on $n - 2$ disks from the first temporary peg to the end peg using the start peg and the second temporary peg as the first and second temporary pegs respectively. Also include the move counting like you did with the three peg version.

Write a main that will ask the user for the number of disks, have the program print out the list of moves for both the 3 peg and 4 peg versions along with the number of moves for each. Runs on 3, 4, and 5 disks are below. In each of these we have the starting peg as A, the ending peg as B and the temporary peg as C for the three peg version and pegs C and D for the four peg version.

```
Enter the number of disks: 3
The solution for n = 3 disks. Using 3 pegs.
Move A to B
```

```
Move A to C
Move B to C
Move A to B
Move C to A
Move C to B
Move A to B
Number of moves in the solution = 7
```

```
The solution for n = 3 disks. Using 4 pegs.
Move A to C
Move A to D
Move A to B
Move D to B
Move C to B
Number of moves in the solution = 5
```

```
Enter the number of disks: 4
The solution for n = 4 disks. Using 3 pegs.
Move A to C
Move A to B
Move C to B
Move A to C
Move B to A
Move B to C
Move A to C
Move A to B
Move C to B
Move C to A
Move B to A
Move C to B
Move A to C
Move A to B
Move C to B
Number of moves in the solution = 15
```

```
The solution for n = 4 disks. Using 4 pegs.
Move A to B
Move A to C
Move B to C
Move A to D
Move A to B
Move D to B
Move C to D
Move C to B
Move D to B
Number of moves in the solution = 9
```

```
Enter the number of disks: 5
The solution for n = 5 disks. Using 3 pegs.
Move A to B
Move A to C
Move B to C
Move A to B
Move C to A
Move C to B
Move A to B
Move A to C
```

```
Move B to C
Move B to A
Move C to A
Move B to C
Move A to B
Move A to C
Move B to C
Move A to B
Move C to A
Move C to B
Move A to B
Move C to A
Move B to C
Move B to A
Move C to A
Move C to B
Move A to B
Move A to C
Move B to C
Move A to B
Move C to A
Move C to B
Move A to B
Number of moves in the solution = 31
```

The solution for n = 5 disks. Using 4 pegs.

```
Move A to D
Move A to B
Move A to C
Move B to C
Move D to C
Move A to D
Move A to B
Move D to B
Move C to A
Move C to D
Move C to B
Move D to B
Move A to B
Number of moves in the solution = 13
```