Recursion Programs

apap10.1

1. Write a recursive function to compute the sum of the integers from 1 to a positive integer passed to the function. The function should have only one parameter, the integer value to sum up to.

2. Compute any real number to any integer power with a recursive function. Assume the real number base and the integer exponent will always be positive.

3. Implement Euclid's GCD algorithm recursively:
GCD (m, n) = m if n = 0 else GCD (n, m % n).

4. Write a recursive function that prints an integer with commas in the right places. Make sure you check for leading 0's inside the number (e.g. 114,007,013). It's OK to have leading 0's in front of the number (e.g. 012,498).

5. The Fibonacci sequence of numbers occurs often in computer science and in nature. The sequence is given by

   1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

   where each number after the first two is the sum of the preceding two numbers. Write a function that will recursively return the nth Fibonacci number. The mathematical definition is as follows:

$$f(n) = \begin{cases} 1 & n = 1 \text{ or } n = 2 \\ f(n-1) + f(n-2) & \text{otherwise} \end{cases}$$

6. Write function *FillRect*, whose header is given on the next page. *FillRect* is given a matrix that represents a white screen on which a single rectangle has been drawn, and the position of a pixel that is inside (not on the border of) the rectangle. If the position is on the screen, then *FillRect* should change all of the matrix elements that represent pixels inside the rectangle from *white* to *black*. Write the function *FillRect* recursively.

   For example:

Screen S	Function call	S after the function call
01234567 0wwwwwww 1ww**bbbb**ww 2ww**b**ww**b**ww 3ww**b**ww**b**ww 4ww**b**ww**b**ww 5ww**bbbb**ww 6wwwwwww 7wwwwwww	FillRect(S,3,4)	01234567 0wwwwwww 1ww**bbbb**ww 2ww**bbbb**ww 3ww**bbbb**ww 4ww**bbbb**ww 5ww**bbbb**ww 6wwwwwww 7wwwwwww

Screen S	Function call	S after the function call
01234567 0wwwwwww 1wwwwwww 2wwwwwww 3wwwwwww 4wwww**bbbb** 5wwww**b**www 6wwww**b**www 7wwww**b**www	FillRect(S,6,6)	01234567 0wwwwwww 1wwwwwww 2wwwwwww 3wwwwwww 4wwww**bbbb** 5wwww**bbbb** 6wwww**bbbb** 7wwww**bbbb**
01234567 0wwwwwww 1wwwwwww 2wwwwwww 3wwwwwww 4wwwwwww 5wwwww**bbb** 6wwwww**b**ww 7wwwww**bbb**	FillRect(S,6,8)	01234567 0wwwwwww 1wwwwwww 2wwwwwww 3wwwwwww 4wwwwwww 5wwwww**bbb** 6wwwww**b**ww 7wwwww**bbb**

```
enum MyColorType {white, black};
typedef apmatrix<MyColorType> ScreenType;

void FillRect(ScreenType & m, int row, int col)
```

Download the program apap10-1-6.cpp. You can use this program for your input and display routines.

7. Solve the "faculty and students problem", in which three faculty and three students must cross a river with a boat holding no more than three people, faculty must never be outnumbered by students either on the bank, in the boat or at any exchange, and at least one faculty must be in the boat at all times. You must use a recursive solution, and for every move you must display who is in the boat and on each side of the shore.

   Your solution should work for any size group (the group size is 2 in the example below). The maximum boat size is always the size of one group. Input the size of the group. Here is what the output for a group size of 2 would look like:

```
Size of group: 2
       Left Bank          Boat          Right Bank
       S: 2 F: 2       S: 0 F: 0       S: 0 F: 0
       S: 1 F: 1       S: 1 F: 1       S: 0 F: 0
       S: 1 F: 1       S: 0 F: 1       S: 1 F: 0
       S: 1 F: 0       S: 0 F: 2       S: 1 F: 0
       S: 1 F: 0       S: 0 F: 1       S: 1 F: 1
       S: 0 F: 0       S: 1 F: 1       S: 1 F: 1
       S: 0 F: 0       S: 0 F: 0       S: 2 F: 2
```