Lab 13 - Recursion

Below are several exercises for you to practice writing your own recursive functions. Within your functions, make sure you comment your code to specify the base case and the recursive function call.

1. Write a recursive function that computes the sum of all numbers from 1 to n, where n is given as parameter.

//return the sum 1 + 2 + 3 + … + n

int sum(int n)

1. Write a recursive function that finds and returns the minimum element in an array, where the array and its size are given as parameters.

//return the minimum element in a[]

int findmin(const int a[], int n)

1. Write a recursive function that computes and returns the sum of all elements in an array, where the array and its size are given as parameters.

//return the sum of all elements in a[]

int findsum(const int a[], int n)

1. Write a recursive function that determines whether an array is a palindrome, where the array and its size are given as parameters.

//returns true if a[] is a palindrome, false otherwise

bool ispalindrome(const char a[], int n)

1. Implement a function that generates all substrings of a string. For example, the substrings of the string "rum" are the seven strings "r", "ru", "rum", "u", "um", "m", ""

**Hint:** First enumerate all substrings that start with the first character. There are n of

them if the string has length n. Then enumerate the substrings of the string that you

obtain by removing the first character.

//returns a string vector containing all substrings of the the //string s

vector<string> generate\_substrings(string s)

1. Write a recursive function that takes as a parameter a nonnegative integer and generates the following pattern of stars. If the nonnegative integer is 4, then the pattern generated is:

\*\*\*\*

\*\*\*

\*\*

\*

\*

\*\*

\*\*\*

\*\*\*\*

//draws the above pattern based on the input

void drawPattern(int n)