RECURSION EXERCISES

1. A palindrome is a word that reads the same forwards and backwards, like “level” or “sees” or “deified”. Write a recursive function that checks whether a single word supplied by the user is a palindrome. Don’t worry about upper and lowercase issues. Here is a prototype:
   ```cpp
   bool is_palindrome(unsigned int a, unsigned int b, const string& s)
   ```
   You may want to #include `<string>`; this library is useful because it contains a function `length()`, which tells you how long the string is.
   Hint: there are two variants of the base case: consider the case for “tot” or “toot”, both of which are palindromes. Do not use any local variables.
   EXAMPLES:
   - “ABCDEFGHIJKLMNOPQRSTUVWXYZ” is a palindrome
   - “ABCDEFGHIJKLMNOPQRSTUVWXYZ” is a palindrome
   - “ABCDEFGHIJKLMNOPQRSTUVWXYZ” is not a palindrome

2. (From Prof. Main’s Web page at [http://www.cs.colorado.edu/~main/projects/chap09c.html](http://www.cs.colorado.edu/~main/projects/chap09c.html)) Write a recursive function that takes an integer and prints it out as a binary one. Here is the prototype:
   ```cpp
   void binary_print(ostream& outs, unsigned int n);
   ```
   The function prints the value of n as a BINARY number to the ostream outs. If n is zero, then a single zero is printed; otherwise no leading zeros are printed in the output. The ‘n’ character is NOT printed at the end of the output. Do not use any local variables.
   EXAMPLES:
   - n=0  Output:0
   - n=4  Output:100
   - n=27 Output:11011

3. (From Prof. Main’s Web page at [http://www.cs.colorado.edu/~main/projects/chap09a.html](http://www.cs.colorado.edu/~main/projects/chap09a.html)) Examine this fractal pattern of asterisks and blanks, and write a recursive method that can generate patterns such as this:

```
*  *
  *
* * * *
  *
  *
* * * * * * * *
  *
  *
  *
  *
* * * * * * * *
  *
  *
  *
  *
  *
* * * *
  *
  *
  *
  *
* * *
  *
  *
* * *
```
With recursive thinking, the method needs only seven or eight lines of code (including two recursive calls). Your method should look like this:

```cpp
void pattern(ostream& outs, int n, int i)
// Precondition: n is a power of 2 greater than zero.
// Postcondition: A pattern based on the above example has been
// printed. The longest line of the pattern has
// n stars beginning in column i of the output. For example,
// The above pattern is produced by the call pattern(8, 0).
Hints: You do not need to check the precondition. Think about how the pattern is a fractal. Can you find two smaller versions of the pattern within the large pattern? Here is some code that may be useful within your method:
// A loop to print exactly i spaces (Eliz’s solution modified this to print 2*i spaces):
for (k = 0; k < i; k++) outs << " ";
// A loop to print n asterisks, each one followed by a space:
for (k = 0; k < n; k++) outs << "* ";
```