4.8 Practice Programs

Sample solutions to these problems will be included by the beginning of 2013 at the following website:

http://www.cs.ucf.edu/~dmarino/ucf/transparency/cop3223/book/

1) Write a program that asks the user for a positive even integer input n, and the outputs the sum 2+4+6+8+...+n, the sum of all the positive even integers up to n.

2) Write a program that calculates the amount of money stored in a bank account after a certain number of years. The user should enter in the initial amount deposited into the account, along with the annual interest percentage and the number of years the account will mature. The output should provide the amount of money in the account after every year.

3) Write a program to take in a positive integer n > 1 from the user and print out whether or not the number the number is a perfect number, an abundant number, or a deficient number. A perfect number is one where the sum of its proper divisors (the numbers that divide into it evenly not including itself) equals the number. An abundant number is one where this sum exceeds the number itself and a deficient number is one where this sum is less than the number itself. For example, 28 is perfect since 1 + 2 + 4 + 7 + 14 = 28, 12 is abundant because 1 + 2 + 3 + 4 + 6 = 16 and 16 is deficient because 1 + 2 + 4 + 8 = 15.

4) Write a program that allows a user to play a guessing game. Pick a random number in between 1 and 100, and then prompt the user for a guess. For their first guess, if it's not correct, respond to the user that their guess was "hot." For all subsequent guesses, respond that the user was "hot" if their new guess is strictly closer to the secret number than their old guess and respond with "cold", otherwise. Continue getting guesses from the user until the secret number has been picked.

5) Write a program to play a game of marbles. The game starts with 32 marbles and two players. Each player must take 1, 2 or 3 marbles on their turn. Turns go back and forth between the two players. The winner is the person who takes the last marble. Your program should prompt each player with a message that states the current number of marbles and asks them how many they'd like to take. Continue until there is a winner. Then your program should print out the winner (either player #1 or player #2.) (Incidentally, if both players play optimally, who always wins? What is their strategy?)

6) Write a program that asks the user to enter two positive integers, the height and length of a parallelogram and prints a parallelogram of that size with stars to the screen. For example, if the height were 3 and the length were 6, the following would be printed:

```
******
 ******
  ******
```

or if the height was 4 and the length was 2, the following would be printed:

```
**
 **
  **
   **
```

7) Write a program that prompts the user to enter a positive odd integer, n, representing the height of a diamond, and print the diamond to the screen. For example, with n = 5, the following should be printed:

```
  *
 ***
*****
 ***
  *
```

8) Write a program that prints out all ordered triplets of integers (a,b,c) with a < b < c such that a+b+c = 15. (If you'd like, instead of the sum being 15, you can have the user enter an integer greater than or equal to 6.) You should print out one ordered triplet per line.

9) Write a game that helps kids practice multiplication. Prompt the user to enter how many problems they want and the maximum value for each operand. Create the appropriate number of multiplication problems, prompting the user to enter the answer to each. Keep track of how many problems they get correct and how much time they take. After the problems are finished, print out the total number of problems solved correctly, the percentage of problems solved correctly and the amount of time taken, in seconds. In the following example, total_sec will equal the amount of time in seconds that the code segment took. This method only works if the segment of code takes several seconds or more.

```
int start = time(0);
// Put code segment here
int end = time(0);
int total_sec = end-start;
```

In order to make the time function call, `#include <time.h>` at the top of your program.

10) Write a program that prompts the user for a positive integer $n \leq 46$ and prints out the n^{th} Fibonacci number. The Fibonacci numbers are defined as follows:

$F_1 = 1$, $F_2 = 1$ and $F_n = F_{n-1} + F_{n-2}$.

The reason the input is limited to 46 is that the 47^{th} Fibonacci number is too large to be stored in an integer. Your program should calculate the numbers the same way one would do by hand, by adding the last two numbers to get the following number in succession: 1+1 = 2, 1+2 = 3, 2+3 = 5, 3 + 5 = 8, etc.