The University of Chicago
Department of Computer Science

CMSC 15200 – Introduction to Computer Science 2
Summer Quarter 2007
Homework #2 (07/27/2007)
Due: 08/01/2007 @ 1:30pm

Name:

Student ID:

Instructor: Borja Sotomayor

Do not write in this area

1	2	3	4	**TOTAL**

Maximum possible points: 75 + 5

The University of Chicago
Department of Computer Science

Exercise 1 <<15 points>>

Write a program that asks the user for an integer number and then displays the following menu:

```
What do you want me to do with this integer?

1. Compute the greatest common divisor with another integer.
2. Compute its natural logarithm.
3. The number is a quantity in degrees. Compute its sine.
4. Exit.
```

When an option is selected, your code *must call a function*. This function can be programmer-defined or part of the standard C/C++ library. Hint: Feel free to reuse code from homework #1.

Exercise 2 <<30 (10+20) points>>

You will write a program that will allow the user to play two popular (yet simple) games: guessing a number and playing the lottery. Your program will start by displaying the following menu:

```
What game do you want to play?

1. Guess a number.
2. Lottery.
3. Exit.
```

Guess a number

If the user chooses "Guess a number", the program will internally think of a number between 0 and 100 and will show the following message:

```
I'm thinking of a number between 0 and 100. Can you guess it?
```

The program will ask for an integer number. If...

- ... the number is larger than the secret number:

  `I'm thinking of a smaller number. Please try again.`
  (the program must now ask the user for another number)

- ... the number is smaller than the secret number:

  `I'm thinking of a larger number. Please try again.`
  (the program must now ask the user for another number)

- ... the user guessed the secret number:

```
Correct! You guessed the secret number!
```

Note: Although the secret number is known to be between 0 and 100, you do not need to validate if the number introduced by the user is within this range (although showing a warning message would be a good idea)

Lottery

If the user chooses "Lottery", the program will ask the user for six integer numbers between 1 and 49 (you must validate these numbers). Next, the program will think of six numbers (again between 1 and 49), and will inform the user of how many numbers he guessed.

Exercise 3 <<10 points>>

Write a program that asks the user to enter a number *x*. The program will then ask the user for *x* numbers, and will then write them in inverse order.

```
How many numbers do you wish to enter? 5
Enter number #1: 10
Enter number #2: 56
Enter number #3: 34
Enter number #4: 5
Enter number #5: 103

Reverse: 103 5 34 56 10
```

Exercise 4 <<20+5 points>>

Write a basic "hangman" game. At this point, we are assuming that you are playing against a friend, so you will first specify a word (your friend must not see you type it in) which your friend will have to guess. More specifically, your program will do the following:

- Ask the user for a word (e.g. "scrumptrulescent")
- Show the user as many hyphens as letters in the word (this is the 'masked word'), and the number of misses left:

```
---------------
Misses left: 6
```

- Ask the user for a letter:

  ```
  Enter a letter: e
  ```
- If the letter is in the word, then show an updated masked word and number of misses left:

  ```
  ----------e--e--
  Misses left: 6
  ```
- If the letter is *not* in the word (let's assume the user typed in 'z'), show a warning, the masked word, and the updated number of missed left:

  ```
  The letter 'z' is not in the word!
  ---------------
  Misses left: 5
  ```
- If the user guesses the word without using up the number of misses, show a congratulatory message. Otherwise, tell him he has lost the game and reveal what the word was.

The next page shows what a run of this program would look like.

Note: If you actually *do* want to play this game with a friend (which would be a fun way of testing that your program works correctly), remember you have to clear the terminal screen after you enter the word (you can do this by writing out an arbitrarily large number of empty lines, although 25 usually does the trick).

Extra credit (5 points): As stated, the hangman program does not keep track of what letters have already been entered. i.e., if a user enters a letter that has already been entered, the number of missed left is decreased (regardless of whether that letter already produced a "miss", or even if the letter was previously guessed correctly). Modify the program so it will keep track of what letters have already been entered. If a user enters a letter more than once, only a warning message it printed out (e.g. "Letter x has already been entered"), and the number of misses is *not* decreased.

```
Enter a word: hatter

------
Misses left: 6

Enter a letter: a

-a----
Misses left: 6

Enter a letter: b

The letter 'b' is not in the word!
-a----
Misses left: 5

Enter a letter: c

The letter 'c' is not in the word!
-a----
Misses left: 4

Enter a letter: d

The letter 'c' is not in the word!
-a----
Misses left: 3

Enter a letter: e

-a--e-
Misses left: 3

Enter a letter: h

ha--e-
Misses left: 3

Enter a letter: t

hatte-
Misses left: 3

Enter a letter: r

hatter: You guessed the word!
```