Introduction to C - Programming Assignment #4

Due Date: October 29, 2009, 11:59pm

Objective

1. To give students practice in using functions

Problem: Educational Software

Your little brother is having trouble with arithmetic. Your parents realize that after taking a few weeks of your C programming course, that you could potentially write a computer program that will allow him to practice his arithmetic skills.

In particular, your program will allow your brother to play two separate games:

1) A game where he has to complete several additions or multiplications.

2) A game where he has to complete several factorizations.

Your program should prompt your brother with the following menu:

1) Play Arithmetic Game
2) Play Factoring Game
3) Print Score
4) Quit

If he chooses option 1, then you should prompt him with the following menu choices:

1) Addition
2) Multiplication

Your program should then prompt him for the maximum value of the numbers to be used in the problems and the total number of questions.

Then it should play the game by prompting him with the desired number of additions/multiplications. Your program should keep track of how much time he takes for these problems. In addition, 5 penalty seconds should be added for each incorrect response. The total time he takes (taking into account the penalty seconds) will be used to determine his score for the round.

If he chooses option 2, your program should prompt him for the desired number of factoring questions and the maximum value of a factor. Your program should keep track of how many guesses he has to make before he gets the correct factorization. (Note: In this game, you will generate two random integers in between 2 and the maximum factor value and multiply them together. This product will be what you show the user. Then the

user will have to guess the original two numbers. Unfortunately, the user may type in two numbers that multiply to the correct value, but aren't the ones chosen by the computer. In this case, you tell the user that their product is correct, but that they haven't guessed the correct factors yet. If their product is just plain incorrect, this should be stated as well.) Each incorrect guess without a matching product is assessed 5 penalty points. Each incorrect guess with a matching product is assess 2 penalty points.

For option 3, simply report your brother's total score, which is the sum of his scores from each round he plays.

Scoring Details

For both of the games, the score your brother earns is equal to the total amount of time (in seconds) it took him to finish the problems (including penalty seconds) divided by the number of problems he solved.

The score in seconds then must be converted to an integer number of points in between 0 and 10. In particular, the conversion works as shown in the chart below:

Time, t, (in seconds)	Corresponding Score
$t < 1$	10
$1 \leq t < 2$	9
$2 \leq t < 3$	8
$3 \leq t < 4$	7
$4 \leq t < 5$	6
$5 \leq t < 6$	5
$6 \leq t < 7$	4
$7 \leq t < 8$	3
$8 \leq t < 9$	2
$9 \leq t < 10$	1
$t \geq 10$	0

Implementation Details

You will be required to write three functions with the prototypes given below. (Note: you may write other functions as well, but these three are required.) Your functions should do what the comments for them below specify:

```
// This function gives the user quantity arithmetic
// questions, where each operand ranges from 1 to max,
// inclusive. The value of operator dictates whether
// the problems are addition or multiplication problems.
// Namely, if op is 1, they are addition problems,
// otherwise, they are multiplication problems.
// The function returns the number of seconds the user took
// to play the entire game, divided by the number of
// problems they solved.
double arithGame(int max, int quantity, int op);

// This function allows the user to play the factoring game
// where 2 randomly generated number lie in between 2
// and max, inclusive. The product is displayed to the
// user. The user must guess the original two numbers
// multiplied to obtain the product. This process continues
// until the user gets both numbers. An error message
// should be given indicating if the product is correct or
// not and if only the values are incorrect.
// The value returned is the number of seconds the user
// took to finish the game including penalty points divided
// by the number of problems.
double factorGame(int max, int quantity);

// Returns the number of points the user has earned based
// on time. In particular, if time is less than 1, 10 is
// returned. Otherwise, if it is less than 2, 9 is
// returned, etc. If time is greater than or equal to 10,
// then 0 is returned.
int numPoints(double timesec);
```

Other Useful Information

Seed the random number generator at the beginning of your program. Do this exactly once. Here is the line of code:

```
srand(time(0));
```

In order to use this you need to include stdlib.h and time.h at the top of the program.

Please use the following constants for ADD and MULT

```
#define ADD 1
#define MULT 2
```

In order to calculate how much time something takes, you can use the time function. In particular, the function call time(0) returns a double that represents the number of seconds after some time. In order to effectively use this, you must call the function twice: once right before you start what you want to time, and once right afterwards. Subtract these two values to obtain the amount of time a segment of code took. Here is a short example:

```
int start = time(0);
// Insert code you want to time here.
int end = time(0);
int timespent = end - start;
printf("Your code took %d seconds.\n", timespent);
```

In order to carry out the scoring function, it may be helpful to look at the following functions in the math library:

```
double ceil(double x);
double floor(double x);
```

Remember, if you want to convert a double to a corresponding integer, you can use a cast as in the example below, where we assume that value is an integer and seconds is a double:

```
value = (int)seconds;
```

References

Textbook: Chapters 9, 10

Notes: Lectures on loops, functions, random number generator functions

Restrictions

Name the file you create and turn in *game.c*. Although you may use other compilers, your program must compile and run using Dev C++. Your program should include a header comment with the following information: your name, course number, section number, assignment title, and date. You should also include comments throughout your code, when appropriate. If you have any questions about this, please see a TA.

Deliverables

A single source file named *game.c* turned in through WebCT.

Sample Output

```
Please make a selection from the following:
1. Play Arithmetic Game.
2. Play Factoring Game.
3. Print Score.
4. Quit.
1

Would you like, 1)Addition or 2)Multiplication?
1
What is the maximum number you would like?
100
How many problems do you want?
4
What is 21+86?
107
Correct, great job!
What is 87+96?
173
Sorry, that's incorrect, the answer is 183.
What is 86+70?
156
Correct, great job!
What is 55+4?
59
Correct, great job!
You took an average of 6.0 seconds per question.
Your score for the round is 4.

Please make a selection from the following:
1. Play Arithmetic Game.
2. Play Factoring Game.
3. Print Score.
4. Quit.
2
Enter the maximum value of a factor.
80
How many factoring problems do you want?
2
Please factor 124.
12 4
Sorry, 12 x 4 does NOT equal 124.
Please try again.
4 31
Sorry, that is not the factorization we are looking for.
Please try again.
2 62
```

```
Correct, great job!
You got the factorization in 3 guesses.

Please factor 24.
12 2
Correct, great job!
You got the factorization in 1 guess.

Great, you took an average of 5.5 seconds per question.
Your score for the round is 5.

Please make a selection from the following:
1. Play Arithmetic Game.
2. Play Factoring Game.
3. Print Score.
4. Quit.
3
Your score is 9.

Please make a selection from the following:
1. Play Arithmetic Game.
2. Play Factoring Game.
3. Print Score.
4. Quit.
4
Thank you for playing!
```

EXTRA CREDIT

Add another menu item to your program called sort. When your brother chooses this option, the program asks him to enter a series of integers (which should be stored in an array) and sorts them by calling a function called selection_sort. When given an array of n elements, selection_sort should

1. Search the array for the largest element and move it to the last position in the array.
2. Call itself recursively to sort the first n-1 elements of the array.