Operating Systems (CS:3620)

Assignment 2 Total points: 100

Due: by 11:59 pm of September 14, 2017

This assignment will contribute 10% to your final grades.

Submission: Please submit your source code through ICON. All your C files should be in the corresponding folder for that specific problem. All of these folders should be in a single folder named "Assignment-2". You will then zip the "Assignment-2" directory and will name the zip file in the following way: *<last_name>-<first_name>-Assignment_2.zip*. You should replace <last_name> and <first_name> with your last and first name, respectively.

Instructions: For each of the following problems, you will need to write a small C file to solve each problem. The name of each C file for each problem is mentioned in the associated problem statement below. For instance, the name of the C file for Problem 2 is "Problem2.c" Please refer back to the files found in ICON under the "Supplementary Material" folder for help.

Please make sure your source code compiles with GCC, otherwise you will not obtain any points.

For compiling a C program named `Problem1.c` in the terminal using `gcc` and in the process generate the executable file `problem1`, type in the following command: `gcc Problem1.c -o problem1`. To run the executable without any command line arguments, you would need to type in `./problem1` in the terminal.

Grading: We will use automated grading scripts with pre-determined inputs and some manual inspection to check each program for correctness. *Please make sure to follow the C file name convention for each problem exactly as directed.* Otherwise, your assignment may not be graded properly. Please make sure you are using an Ubuntu environment for this assignment. We will provide a Ubuntu virtual machine image. Please see the end of the assignment for detailed instructions about setting up the virtual machine. Note: This is the same image as Assignment 1. **If you successfully installed this image, there is no need to go through the installation process again.**

Cheating and Collaboration: ***This is an individual project****, you can discuss with your peers but cannot copy source code. Please do not copy source code from Internet. Think about the worst-case scenario when you get caught.*

Questions

1. (2.5+2.5=5 points) **Loop Pyramids:** The problem is divided into the following two parts.

   (a) For this problem you will print a pyramid with r rows. Your program should read the value of r from the standard input. The pyramid will be printed to standard output.

   For instance if the user chooses $r = 4$, the expected output is the following.

   <newline> represents the new line character.

   ```
   *<newline>
   **<newline>
   ***<newline>
   ****<newline>
   ```

   Assume the following will hold: $1 \leq r < 80$.

   Please name this C file "`Problem1_a.c`"

   (b) This problem is almost identical to **part a**. Your program should read the number of rows r from the standard input and should print a reverse pyramid in the standard output. The only difference from part (a) is the pyramid shape.

   For instance if the user were to choos r=4, the expected output is the following. **<space> and <newline> represent the space and the new line characters, respectively**.

```
<space><space><space>*<newline>
<space><space>**<newline>
<space>***<newline>
****<newline>
```

Assume the following will hold: $1 \leq r < 80$.

Please name this C file "`Problem1_b.c`"

2. (10 points) **Palindrome:** A palindrome, in our context, is a word that reads the same even when it is read backwards. For instance, `radar, mom, noon`, and `kayak` are all palindromes whereas `homework`, `assignment`, and `operating` are not. For this problem, you can safely assume that words will not contain the newline or space characters but can contain other printable ASCII characters.

   The maximum length of a word will be 1024. You will read your inputs from the standard input and print all your outputs in the standard output.

   The first line of your program's input is N (i.e., $1 \leq N \leq 100$) which indicates the number of words for which you would need to decide whether that word is a palindrome. Then N lines will follow where each of the N lines will contain a word you need to test. If your word is a palindrome you should print "`PALINDROME <newline>`" (without the quotes) whereas print "`NOT<space>PALINDROME<newline>`" (without the quotes) when the word is not palindrome.

   Please name this C file"`Problem2.c`"

3. (10 points) **File Size:** Write a C program that will take a file name f as a command line argument and will display in the standard output the size of f in bytes . You can safely assume that file names are C-type strings (with a null character at the end) of maximum length 32.

   For this problem, you can neither use the `ftell` nor the `fseek` functions.

   Please name this C file"`Problem3.c`"

4. (10 points) **Max/Min/Avg/Med:** Write a C program that calculates the maximum (Max), minimum (Min), average (Avg), and median (Med) values of a sequence of integers. For this problem, you will take input from the standard input and display your program's results in the standard output.

   The first line of the input is an integer T indicating the number of test cases your program should handle. T test cases follow. Each of the T test cases starts with the value N indicating the number of integers in the sequence. N integers would follow. Your program should print the following information (in the order of their appearance): the maximum value among the N integers, the minimum value among the N integers, the average of the N integers, and the median of these N integers.

   Max, Min and Median should be of the **int** type while the average should be of the **float** type.

   Sample input: <newline> represents the new line character.

```
1<newline>
3<newline>
10<newline>
17<newline>
4<newline>
```

   Expected output for the sample input:

```
Max:<space>17<newline>
Min:<space>3<newline>
Average:<space>10.333333<newline>
Median:<space>10<newline>
```

When printing out the average value, please do **NOT** round up. Simply print out the value in the default format.

Please name this C file"`Problem4.c`"

5. (5 points) **Printing Integers in the Ascending Order:** Write a C program that prints a list of input integers in the ascending order. Your program should take input from the standard input and the output should be displayed in the standard output.

   The first line of the input is T representing the number of test cases your program should handle. T test cases will follow. Each of the T test cases will start with a line containing the integer N indicating the number of integers in the list. Then N integers would follow; one in each line. Your program should print these N integers in the ascending order for each test case.

   Sample input:

```
2<newline>
2<newline>
3<newline>
1<newline>
4<newline>
3<newline>
4<newline>
1<newline>
2<newline>
```

   Expected output of the sample input:

```
1<space>3<newline>
1<space>2<space>3<space>4<newline>
```

   Please name this C file"`Problem5.c`"

6. (5 points) **Fibonacci Number:** Write a C program that **recursively** calculates the Nth Fibonacci number. See `https://en.wikipedia.org/wiki/Fibonacci_number` if you do not know what is a Fibonacci number. Your program should take input from the standard input and display its output in the standard output.

   The precise input format is as follows. The first line of the input will contain the integer T indicating the number of test cases your program needs to handle. T test cases will follow. For each of the T test cases, the value of N will be given. Your program should print the Nth Fibonacci number. You can assume the value of N to be small enough so that the Nth Fibonacci number will fit into a `unsigned long long int` (8 bytes).

   Sample input:

```
3<newline>
1<newline>
8<newline>
6<newline>
```

   Expected output for the sample input:

```
1<newline>
21<newline>
8<newline>
```

   Please name this C file"`Problem6.c`"

7. (5 points) **GCD:** Write a C program to **recursively** calculate the Greatest Common Divisor (GCD) of two positive integers. The GCD is the largest positive integer that is able to divide each of the given integers. For instance, the GCD of 8 and 12 is 4. For more details on GCD, please see `https://en.wikipedia.org/wiki/Greatest_common_divisor`.

   Your program should take input from the standard input and print the output in the standard output. The precise format of the input is given below. The first line of the input is an integer T indicating the number of test cases your program should handle. T test cases will follow. For each of the T test cases, the input will contain two space-separated positive integer n_1 and n_2 such that n_1 and n_2 will both fit into a 32-bit integer (i.e., `int`). The output for a particular n_1 and n_2 is their GCD followed by a newline.

   Sample input:

```
2<newline>
8 <space> 12 <newline>
27 <space> 18 <newline>
```

   Expected output of the sample input:

```
4<newline>
9<newline>
```

   Please name this C file "`Problem7.c`"

8. (10 points) **Reading Binary File and Addition:** Write a C program that reads a N number of integer pairs n_1 and n_2 from a binary file f and for each of the N pairs n_1 and n_2, display their summation (i.e., $n_1 + n_2$) followed by a newline character in the standard output. Your program should take the name of the binary file f as a command line argument. The first 4-bytes of the binary file constitutes the value N; the number of integer pairs to follow. For each of the N integer pairs, display their summation in the ASCII format in the standard output. Each element of the pairs is represented as a 4-byte integer.

   Please name this C file "`Problem8.c`"

9. (10 points) **Number Addition:** Write a C program that reads N number of integer pairs n_1 and n_2 from an ASCII file f_{in} and for each of the N pairs n_1 and n_2, store their summation (i.e., $n_1 + n_2$) in a binary file f_{out}. Your program should take the name of the binary files f_{in} (argument 1) and f_{out} (argument 2) as command line arguments.

   The first line of the f_{in} file contains the value N and N pairs of integers n_1 and n_2 follow. For each such pair of n_1 and n_2, store the value of $n_1 + n_2$ in the binary file f_{out}. We do not need to print newlines after each summation.

   Please name this C file "`Problem9.c`"

10. (30 points) **Big Integer Addition:** *Int* datatype can hold an integer of size 4 bytes, *long long int* can hold up to 8 bytes however sometimes the number required for calculations exceeds this. For example, often while doing cryptographic operations (e.g., encryption, decryption) you are required to `add` and `multiply` integers that are much bigger than standard libraries can support.

    Your goal is to implement the addition operation of big integers and print the result to standard output. **These integers WILL NOT fit in a typical primitive data type**.

    You'll be required to store these integers in a different form. For example, you can store these in a char array where each digit in such numbers occupy one position in that array.

    You will be provided a skeleton C file (Inside the "Problem 10" folder). Your objective is to finish coding the "add_and_print" function which takes care of the addition and prints the results to the standard output.

    Your program should take the name of the input file as a command line argument. Just like the following

```
shell% ./Problem10 <INPUT FILE>
```

Sample input file:

```
39589345934538593845<space>3985930495839485<newline>
3945839084598<space>39485930854983485<newline>
39458390845981<space>1000000000<newline>
3349859839845983349593485<space>394589348598345<newline>
```

Expected output for the sample input file:

```
39593331865034433330<newline>
39489876694068083<newline>
39459390845981<newline>
3349859840240572698191830<newline>
```

Please name this C file "`Problem10.c`"

Virtual Machine Instructions:

For this assignment an Ubuntu image has been provided. Please see "**Instructions**" for links and instructions on how to set everything up.

Instructions

1. Please download and install **VirtualBox** from the following link:
   https://www.virtualbox.org/wiki/Downloads
2. Please download the **sec.ova** images from the following link:
   `https://www.dropbox.com/sh/hpgt3tuyev8hlt6/AADkefjEtdzp9mP4YXU1DvFNa?dl=0`
3. Run **VirtualBox**
4. Go to **File/Import Appliance**
5. In the *Appliance to import* window choose *sec.ova* image file and press **Continue**
6. Press **Import** and then the import process will begin
7. After the *Import* process finishes you can see the imported virtual machine (VM) on the left-side window. Select the VM and then press **Start** in the toolbar.
8. Whenever prompted for username and password use the following credential
   - **Username:** sec
   - **Passowrd:** ces
9. After logging in. Go to ICON from the VM and download Assignment 2 found in ICON.