System Skill Midterm Quiz

Date: Thursday, June 9th, 2022
Due Date: Friday, June 10th, 2022 at 11.59PM
Instructor: Rachata Ausavarungnirun

Problem 1 (20 Points):	
Problem 2 (35 Points):	
Problem 3 (35 Points):	
Problem 4 (20 Points):	
Total (100+10 Points):	

Instructions:

1. **DO NOT CHEAT.** If we catch you cheating in any shape or form, you will be penalized very heavily based on **the following plagiarism policy** ($N*$ 10% of your total grade, where N is the number of times you plagiarized previously). This include asking the questions online, copying codes from the internet, etc.
2. If you get 100, you get a full score. Any points above 100 goes to your extra credit.
3. Submit your work as a zip file on Canvas.
4. Because everyone has a lot of time, I expect everyone to **test your code**.
5. If not specified, input and output types are a part of the question. Please use appropriate input and output types that make sense for the purpose of the question.
6. Please clearly comment your code, especially if your code do not work perfectly.
7. Clearly indicate your final answer for each conceptual problem.
8. Your `code should not have any memory leaks.`

Tips:

- **Read everything.** Read all the questions on all pages first and formulate a plan.
- **Be cognizant of time.** It is a sad day if you click submit when the submission site close.
- **Canvas allows resubmission.** I will take a look at the last version you submit.
- **Show work when needed.** You will receive partial credit at the instructors' discretion.

1. Linux Scripting [20 points]

Write the Linux or git scripts to handle the following tasks, please explain your answer:

(a) You are working on the new group assignment. Let's assume that there is a starter code in on github at `https://github.com/rausavar/MUIC-s2021-t3-syskill` (Note: There is nothing there, but just assume there is a repo there). Your task is list the commands to 1) to clone the repository and 2) from the master branch that you just cloned, fork your own branch with your name as the branch name. (5 points)

(b) While you are implementing your exercise, you lost track of the files and the line number of all the codes your friends put in are. Use the grep command to search for the function called "myMalloc" that is located somewhere in your just cloned git repo. You can assume that the repo you cloned is in `./assignmentX`. (2.5 points)

(c) Now, your friends just committed a new set of code, you pull and found that it is wrong. Let's assume that the commit ID of the correct code are at adbe182fe. Type in the command so that your code is reverted back to that version and then push this change so that everyone can see the updated version. (2.5 points)

(d) What does the following regular expression match to: `a.b.*` (2.5 points)

(e) Proivde a regular expression that matches a positive Integer (2.5 points)

(f) List out the commands for the task in our in-class exercise 2 assuming that you are in the group with nothing in your home directory. Explain your reasoning. (5 points)

2. Basic C [35 points]

(a) Write a function called `int isFib(int n)` that returns 1 if the number `n` is a fibonacci number and 0 if not. **You can only use one loop with no recursion for this question to get a full credit**. Otherwise, you will only get 8 points out of 10 points. Save this in `isPrime.c`. (10 points)

(b) In this next question, you will write a function that is used in many modern-day applications called matrix-vector multiplication that multiplies a matrix with a vector (see the definition here `https://mathinsight.org/matrix_vector_multiplication`).

Your task is to write a function called `void matrixVector(int **inMatrix, int *inVector, int * result, int sizeX, int sizeY)` that takes in a matrix `inMatrix` of size `sizeX` by `sizeY`and a vector `inVector` of size `sizeX`. The function should store the result in the resulting vector `result`. Save this file in `matrixVector.c`. (25 points)

Hint: Feels free to start from our sample code that play with matrices.

3. Linked List Strike Back [35 points]

For this question, please use our linked list example code we used in class. This is located in `/sample_code/2021-t3/lecture10-pointer-cont`

(a) From our example code, instead of a linked list storing an integer, we want to store a string in a `char*` format. Rewrite our struct definition and all existing functions from our sample code to handle this change. You can assume that the space for the input string has been properly allocated (10 points)

(b) Then, write a tester to test your code above in the main function (5 points).

(c) Then, write a function pointer called `compPtr` and a function to compare two strings together by summing up all the ASCII value for each characters in the string called `int compFunc(char* a, char * b)`. This compare function will return `-1` if the ASCII sum of `a` is lower than the ASCII sum of `b`, `0` if the ASCII sum of `a` is equal to the ASCII sum of `b` and `1` if the ASCII sum of `a` is greater than the ASCII sum of `b`. Feels free to use string.h and please make sure to use the correct type for our function pointer. (10 points)

(d) Then, write a function called `int countPop(char * input)` that counts the number of string in our linked list that has the ASCII sum greater than `input` (10 points).

4. Data in Each Bytes [20 points]

In this question, you are going to assume the following data type and its corresponding sizes: `char`: 1 byte and `int`: 3 bytes.

From a C-like snippet of a code below, answer the rest of this questions. **Note:** use a lost of casting in the second half of this snippet to make sure things are clearly declared.

```
typedef struct type1{int i[3];} typeA;
typedef struct type2{typeA j[2];} typeB;

typeA *a;
typeB *b;
unsigned long *c;
int i;
a = malloc(2*sizeof(typeB));
b = a;
c = a;
for(i=0;i<18;i++)
{
    *(c+i) = (unsigned char)(i%8);
}
```

Assuming that the actual location of `a` after `malloc` is called is at the physical address `0x10000` in DRAM. Using the concept of address we learn from our class, the table below shows our DRAM starting from the address `0x10000`. **Please put in the values inside each bytes in our DRAM starting at address `0x10000` up to the end of where `a` is malloced for. Write `XX` in the byte that is unknown/unassigned, and write `YY` on the byte that is not in the range that `a` covers**. I gave you the first byte, finish the rest.

Hint: Draw the organization of the struct out on a piece of paper. Pair this up with the concept of an array and type casting. This question is actually very easy but you need to be a bit careful.

Hint2: Please note that our table is actually 1D. Each row is 8 bytes in size and the address of the next row basically continue from the end of the earlier row. I just have no method to draw a very long 1D table so I need to resort to a 2D table here.

Initials: ______________________________

Base Address	**Base+0**	**Base+1**	**Base+2**	**Base+3**	**Base+4**	**Base+5**	**Base+6**	**Base+7**
0x10000	0							
0x10008								
0x10010								
0x10018								
0x10020								
0x10028								
0x10030								
0x10038								
0x10040								
0x10048								