CSE 3100 Systems Programming Spring 2024

Practice Exam #1

This is a 110-minute exam. *Please read every question carefully.*
Please read the following list of important items on exam policy and make sure you understand it.

- The Makefile and starter code for the exam can be downloaded from HuskyCT.
- During the exam, you cannot communicate with and/or obtain help from people other than TAs and instructors on exam questions. Particularly, you cannot use any messaging applications.
- During the exam, you can only access data/files on HuskyCT, your Virtual Machine, man pages and files on your own laptop and on the lab computer you use. You cannot access data/files on other computers/servers. You cannot use your phone. You cannot use Chat-GPT or other generative AI.
- You must submit your code on Gradescope. We only grade submissions we have received on Gradescope.
- Note if your code does not compile, you will lose at least half of the points for that question.
- Do not modify the starter code.
- Submitting to Gradescope after leaving the exam room is not allowed.
- Return the question paper before you leave the exam room.
- After the exam, you cannot discuss/disclose exam problems and/or share your code with students who have not taken the exam.

Full Name: _____ NetID: _____

Lab Section: _____ Date: _____

Signature: _____

Question 1. (30 points) Sort array of Squares

Given an array of integers **arr** sorted in ascending order, the task is to sort the squares of given integers. There are many ways to achieve this, but we are given a sorted array to use to our advantage.

We use two variables, **left** pointing at the beginning of the array and **right** at the end of the array. We will use a temporary array **result** to store the squares in ascending order before we copy the sorted squares to **arr**.

Iterate through the **result** array from the last index to store the square of the largest integers from the end. First, we compare absolute values of **arr**[**left**] and **arr**[**right**];
If **arr**[**left**] > **arr**[**right**], we store the square of **arr**[**left**] in the **result** and move the **left** variable to the next index.
If **arr**[**left**] < **arr**[**right**], we store the square of **arr**[**right**] in the result and move the **right** variable to the next index. Finally, we copy all the elements of **result** to **arr**.

Below is the starter code for this question. (squares.c in Husky CT). Do not use math library functions.

```
#include<stdio.h>
#include<stdlib.h>

void print_array(int *arr, int n){
    //do not modify this function
}

int absolute(int a){
    //fill code here
    // function should return absolute value of a number absolute(9)=9, absolute(-9)=9
}

void sort_squares(int *arr, int n){
    int left = 0;
    int right = n - 1;
    int *result = (int *)malloc(n * sizeof(int));

    for(int i = n - 1; i >= 0; i--){
        if(absolute(arr[left]) > absolute(arr[right])){
            //fill code here
        }
        else{
            //fill code here
        }
    }
    // copying elements of result to arr
    for(int i = 0; i < n; i++){
        arr[i] = result[i];
    }
    //fill in the code
}

int main(int argc, char *argv[]) {
 //Do not modify main function
}
```

Below is the test case for this question. Submit squares.c to gradescope.

```
$./squares -9 -4 -1 0 2 6
0 1 4 16 36 81
```

Question 2. (40 points) Addition and subtraction of two large numbers

Given two non-negative large numbers, **num1** and **num2** (at least 30 digits and num1 > num2), we are tasked to find the sum and difference of those two numbers.

We represent each number using a linked list with each digit in a node. We use **prepend()** function from linkedlist.c to store digits in a list.
123 is represented as 3⟶2⟶1 with the head pointed towards the node containing 3.

The function **addition()** takes two pointers pointed at the head of the numbers and returns the sum of the numbers. We should loop through both lists, calculate the sum of individual digits, add carry if it exists, and store the sum of individual digits in the new list result.

The function **subtraction()** takes two pointers pointed at the head of the numbers, subtracts num2 from num1, and returns the difference. We should loop through both lists, find the difference, borrow from the next digit if needed, and store the difference of individual digits in the new list result.

All helper functions required to handle linked list operations are given in linkedlist.c. Structure of node is given in linkedlist.h.

Below is the starter code for this question (calculator.c in husky CT, also needs linkedlist.h and linkedlist.c)

Do no modify or add new functions to linkedlist.c and linkedlist.h files. Any new helper function should only be defined in the calculator.c file. (Ask your TA if you have any questions.)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#include "linkedlist.h"

node* addition(node* num1, node* num2) {
    node* result = NULL;
    int carry = 0;
    //TODO: Write code to add two numbers represented by num1 and num2

    return result;
}

node* subtraction(node *num1,node *num2){
    node *result = NULL;
    int borrow = 0;
    //TODO: Write code to subtract num2 from num1

    return result;
}
```

```
node *remove_leading_zeros(node *num){
 //Do not modify this function
}

int main(int argc,char *argv[]){
 //Do not modify main function
}
```

Below is the test case for this question.

```
$./calculator 12345678910111213141516171819202345678910111213141516171819201469135782022242628303234363840
100000000000000000000000000000000
```

Submit calculator.c to gradescope.