COSC 350 System Software Midterm #2-1

04/14/2020

Name:_____.

1. (15 pt.) Write a following error free C program (**DO NOT USE GLOBAL VARIABLE**)
   A child process sends the message "Hi, Mom" over a **pipe** to its parent process. The parent process reads the message and prints it to standard output as “my child said Hi Mom”. Then parent process sends the message “I love you” over a **pipe** to its child. The child process read message and prints it to standard output as “My mon said I love you”. Assume that all system calls succeed (no need to error check).

2. (20 pt.) Write following syntax error free compliable program “twosum.c”.

   anyinput.c

   following program get any data from standard input and write on standard output. This program keeps running until Ctr-D (no more data)

```
//anyinput.c
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define MAXLINE 256

int main()
{
    char rline[MAXLINE];
    int  size;


   while ( (size = read(STDIN_FILENO, rline, MAXLINE)) > 0)
   {
         write(STDOUT_FILENO, rline, size); /*write to a pipe */
   }
   return 0;
}
```

twosum.c (**DO NOT USE GLOBAL VARIABLE)**

popen() internally do followings by system

- creating a pipe
- forking a child,
- closing the unused ends of the pipe,
- call exec and the child execute a program (child can read or write data though the pipe).

By using **popen**, let a child runs previous program (**anyinput**) and send data to parent through a pipe. If inputs are two integer values, calculate multiplication of two numbers and write the result on standard output. If inputs are not two integer values, it must respond as "**invalid inputs: two integers**". This program keeps runs until the child process 's (anyinput program) termination. (We assume two programs **twosum** and **anyinput** are located under same directory)

3. (15 pt.) Write following syntax error free compliable program. (**DO NOT USE GLOBAL VARIABLE**) Create three threads and each thread runs its own function fun1, fun2 and fun3 respectively.
   - The first thread gets a command line integer value (from argv [1]) and calculate +10.
   - The second thread add 20 to the result of the first thread.
   - The third thread add 30 to the result of the second thread.
   - Then, original program prints the final result of calculations as **"The three thread's calculation result is …."**

   If we run this program with command line input 20 (./task3 20), output of original program will display

   "The three thread's calculation result is 80"