COMP 206
Assignment #3

Due: March 21, 2017 on MyCourse by 23:30

This assignment builds on the NewProject script you wrote in A2.

Using that script, you will create the necessary directories and compile the programs in question 1 and 2. You will also use NewProject for question 3, however you will be asked to modify NewProject so that it will use a makefile.

Question 1

1. Create a program named **prog1.c**.
   You are given a small text file to test with your program.

   Read the contents of the file and output the following:

   i. The sum of the numbers in the file

   ii. The Fibonacci sequence of each number in the file. You should use an **iterative algorithm** to compute the Fibonacci numbers. Specifically use Algorithm 2 in the link below.

   https://www.ics.uci.edu/~eppstein/161/960109.html

   A sample input text file (input.txt) is provided with this assignment, in grading this question, we would test with different test cases but with the same file format.

   Given an input file containing "1 2 3 4 5"; Your output will look as follows:

   The sum of numbers is 15

   The Fibonacci Sequence of each number is:

   Fib(1) is 1

   Fib(2) is 1

   Fib(3) is 3

   Fib(4) is 5

   Fib(5) is 7

Question 2

2. Write a program **prog2.c** that implements versions of the library function *strncpy*, *strncat* and *strncmp*. Your program takes three arguments char *s, char *t, and int n. Your program operates on, at most, the first n characters of its argument string. In the case of strncpy, it copies at most n characters from t to s. The program asks for the input from the user from the main function.

   N.B : You are to use **pointers** to implement this exercise.

Sample Input and Output

./prog2

Your output should be

Enter the first string: "I am a good student "

Enter the second string: "Noah also is a good student"

Enter the number: 12

strncpy is "Noah also is"

strncat is "I am a good student Noah also is"

strncmp is "I am a good student " < "Noah also is a good student"

Question 3

3. For this program you are expected to build your code with a makefile. Your NewProject Bash script generates the required makefile. The c programs that you are to compile with the makefile are the files: lib.c (which contains 2 functions, fib() and sort()) and prog3.c (which contains your main() function described below).

   To build your code, use the following commands

   ./NewProject path project_name file_names

   make

   NewProject behaves almost the same as from assignment #2, except that we have added file_names list. For example: ./NewProject 206Ass3 q3 prog3.c lib.c. The program will CD into 206Ass3 and create the directory q3, and in there it will create the source subdirectory. In the source subdirectory it will create the makefile using the file_list (prog3.c and lib.c) but it will also assume .h files for each file. The user will then use the "make" command to compile the program. See the class slides for the makefile structure.

   Write a program **prog3.c** that takes 2 positive integers, x and y from the command line and creates 2 child processes using fork(), waits for the two child processes to complete, and then terminates. The first process sorts x random integers using any sorting algorithm (write your own sort algorithm) and the second process computes the y-th Fibonacci number F(y) using recursion. These two functions are contained in the lib.c file.

   Here are the details:

   i. Implement your own **sort** algorithm

   ii. The y-th Fibonacci number is defined recursively as:

   F(1)= F2 = 1

   F(y) = F(y-2) + F(y-1) if y > 2

   Use the **recursive formula** above to compute the y-th Fibonacci number. Use Algorithm 1 in the link below.

   https://www.ics.uci.edu/~eppstein/161/960109.html

Specifications

1. Your program is run with the following command line

   ./prog3 x y where x and y are positive integers

2. The main program creates two child processes, and waits for their completion, and then exits.

3. The first child generates x random integers with C function **srand()** and **rand()** into a local array of the child process. Each number should be in the range of 0 to 99. You can easily use r**and() % 100** to put the random numbers within the range. The process should do the following in this order:

   i. Prints the random generated integers

   ii. Use your sort algorithm to sort the array

   iii. Prints the sorted array.

4. The second child process computes the y-th Fibonacci number F(y) with **recursion.** The process should do the following in this order:

   i. Prints the value of y

   ii. Use the recursion to compute f(y)

   iii. Prints the result

   Use **long** for computation.

Sample Input and Output

./prog3 8 10

Your program should generate 8 random numbers and sort them, then compute the 10-th Fibonacci number.

Output from the first process (sort) should be

Heap Sort Process Started

Random Numbers Generated:

4 29 1 56 34 27 93 45

Sorted Sequence:

1 4 27 29 34 45 56 93

Heap Sort Process Exits

N.B: Use %4d to print your integers and all output from this process should start on column 4 (i.e., 3 leading spaces)

Output from the second process (Fibonacci) should be

Fibonacci Process Started

Input Number: *10*

Fibonacci Number f(10) is *55*

Fibonacci Process Exits.

N.B: Since the computed Fibonacii number may be very large, use %1d to print your computed result and all output from this process starts on column 7 (i.e., 6 leading spaces)

The main program should print the following output:

Main Process Started

Number of Random Numbers = 8

Fibonacci Input = 10

Heap Sort Process Created

Fibonacci Process Created

Main Process Waits

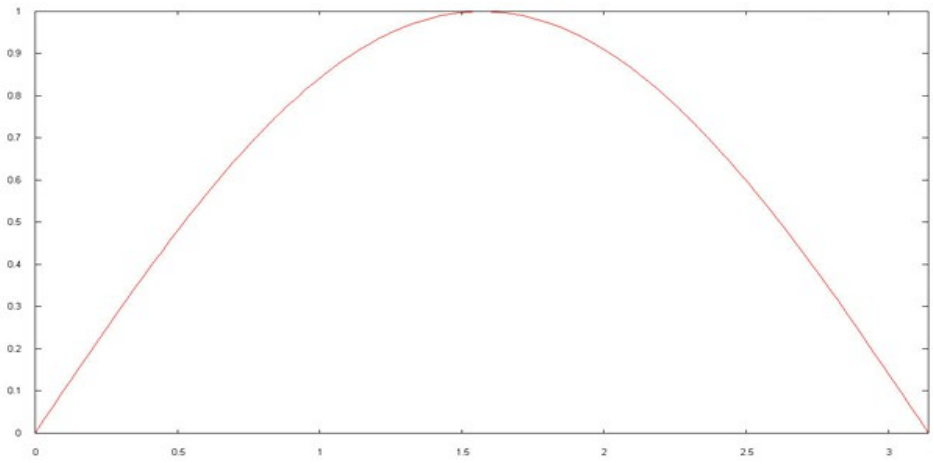
Main Process Exits

N.B: It is very important to remember that both process **MUST** run concurrently and may print their output lines in an unpredictable order. As a result, the output lines from both processes may mix which is why you need proper indentation to know which process prints what. Also make sure that each line does not contain output from different processes.

For the Glory

Write a **prog4.c** that builds on Q3. You are required to create 3 processes, the first and second process remain the same in the specification given in Q3. The third process computes the integration of function of sin(x) between 0 and π.

Integration as given in your calculus class is:



This means that the area between the sin (x) function and the x-axis is 2. The area is completely enclosed in a rectangular area with length π and height 1 as shown in the figure below. The area of this rectangle is π*1 = π. You may use **acos(-1.0)** to obtain the value of π.

If we randomly pick s points in the rectangle and find out *t* of them in the area of the sin(x) function, then the ratio t/s suggests that the area under sin(x) is t/s of the rectangle. Therefore, (t/s)* should be close to 2. Here is what your program should do.

1. Generate two random numbers a and b where 0 <= a < π and 0 <= b < 1. This (a,b) represents a point in the rectangle.
2. If b <= sin(a), point (a,b) is in the area between sin(x) and the x-axis.
3. Let the total number of points in the area between sin(x) and the x-axis be t.
4. Then, (t/s)* π should be close to 2, the desired result. This is especially true for large s.

Your child process simulates the calculation of the integration of the sin(x) function as given earlier and it does the following work.

1. Prints the value of S.
2. Iterates s times and count the number of points in the area between sin(x) and the x-axis.
3. Prints the computed area.

It should have the following input:

./prog4 8, 10, 200000

1st and second process runs as specified in Q2 and third process output is specifically:

Integration Process Started

Input Number 200000

Total Hit 127498

Estimated Area is 2.0027339

Integration Process Exits

N.B: All output from this process starts on column 10 (i.e., 9 leading spaces).

Note that the process results might be slightly different from the output shown here even with the same s because random numbers are used.

The main program should print the following output:

Main Process Started

Number of Random Numbers = 8

Fibonacci Input = 10

Integration Iterations = 200000

Heap Sort Process Created

Fibonacci Process Created

Integration Process Created

Main Process Waits

Main Process Exits

Modify your NewProject Script to include a CMakeLists.txt to compile Q4.

General Submission Guidelines

1. All programs must be written in C
2. Your programs should be named prog1.c, prog2.c, prog3.c and prog4.c
3. You should submit your NewProject script from Assignment 2 which would be used to compile your Q1, Q2.
4. You should submit your modified NewProject2 script file which would be used to compile your Q3 and the optional "For the Glory".
5. Do not submit any object or executable files

Grading Guidelines

All questions are graded proportionally.
The assignment is graded out of 20 points.

Question 1 – 5 points
- +2 Text file
- +1 Sum
- +2 Fibonacci

Question 2 – 5 points
- +1 Main
- +1 strncpy
- +1 strncat
- +2 strncmp

(Question 3 on next page)

Question 3 – 10 points

- +2 NewProject2
- +2 Sort
- +1 Random
- +1 Fib
- +7 Fork()
- +2 Command-line arguments
- +5 makefile