CECS 326

Spring 2017

Haney Williams

Project 2

Phanna Chrin

Amanda Pan

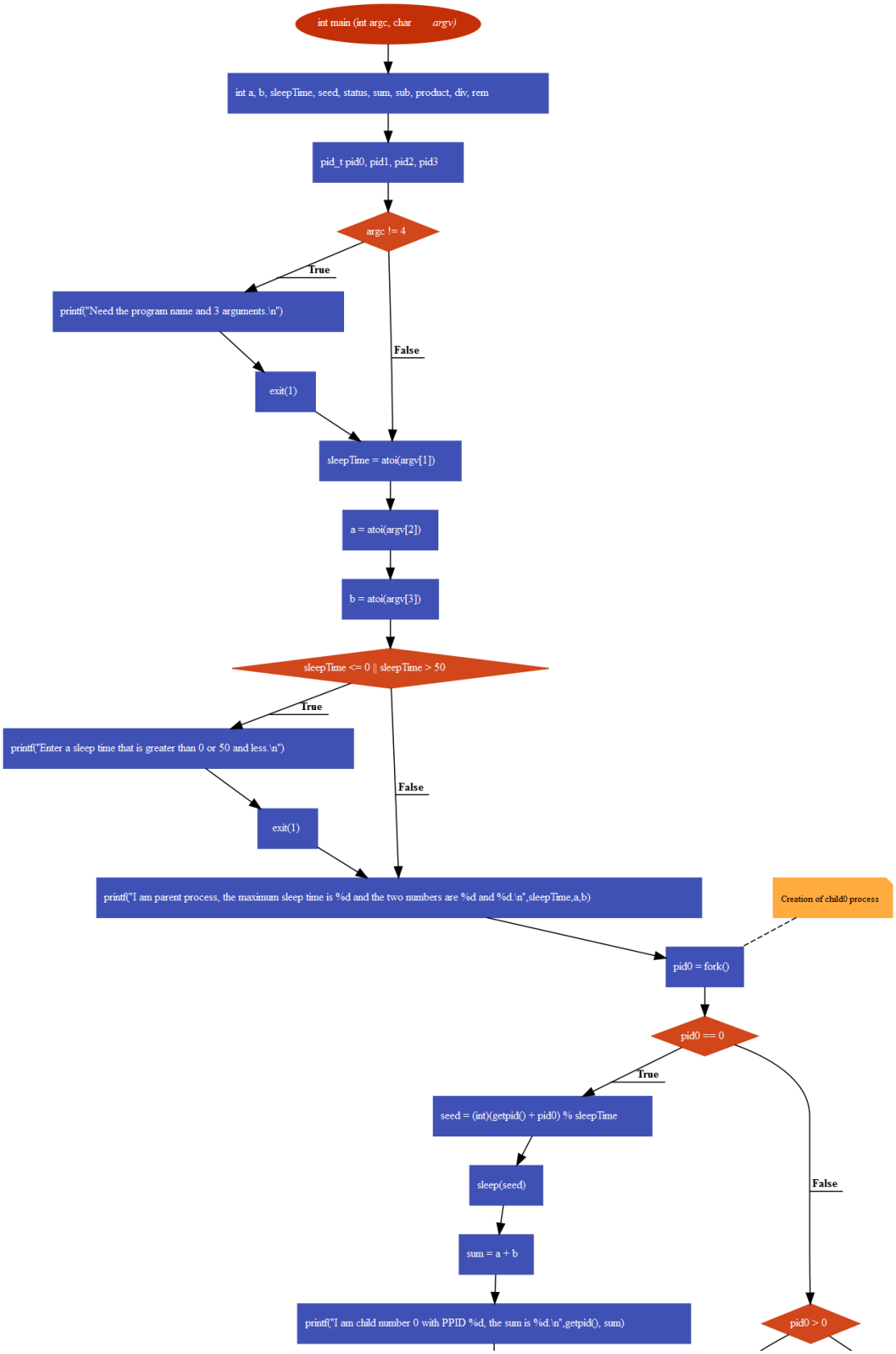
Given Problem:

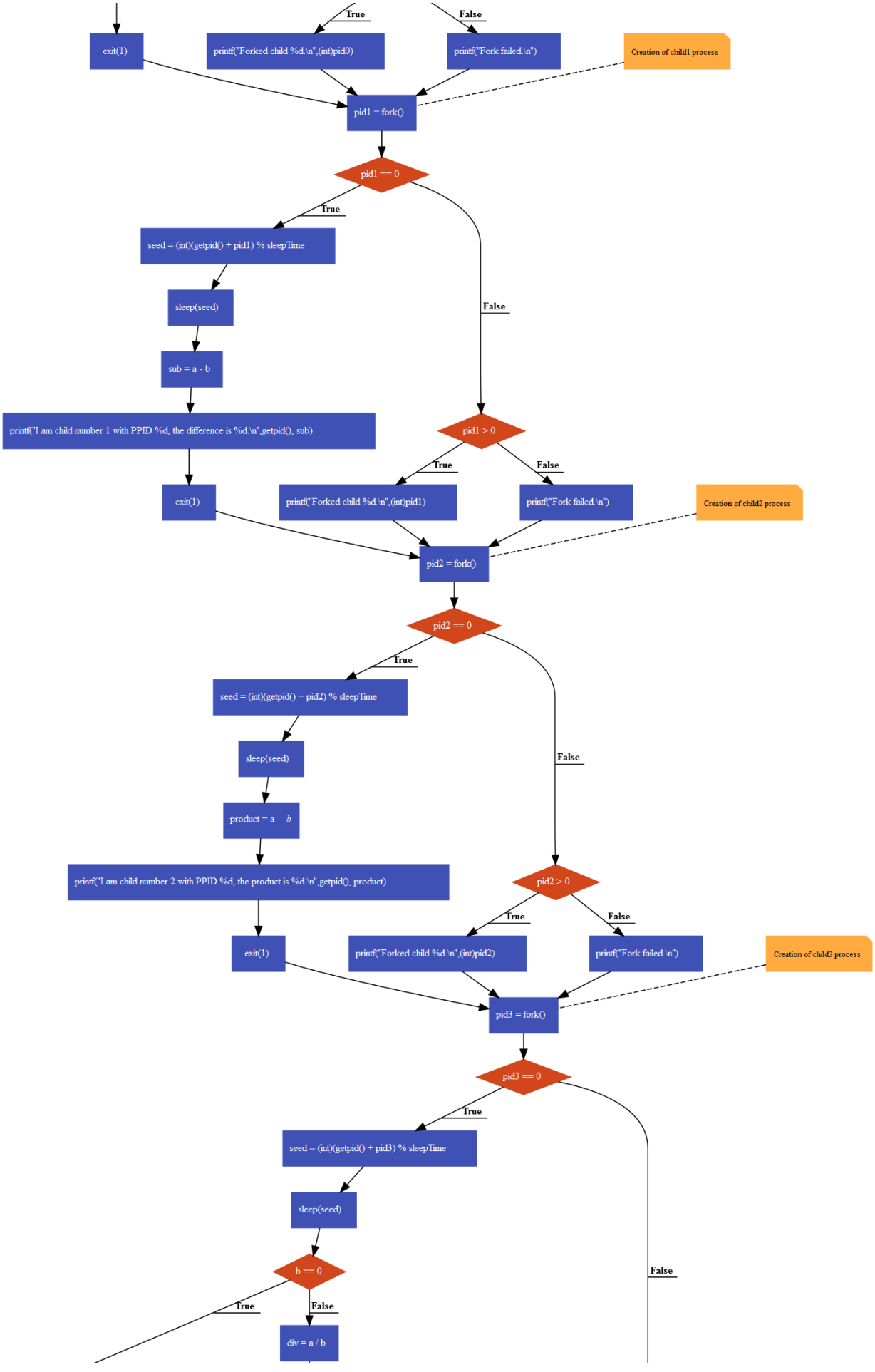
Write a parent program and a child program using C/C++ to do the following. The parent program uses a command line argument to generate 4 child processes. The parent program receives two integer numbers A and B and a sleep time T from the command line arguments. The parent displays the two numbers, sleeps for a random time modulo T, then passes these two numbers together with the child number and the sleep time T to the 4 child process. The child program sleeps for a random time modulo T, then performs an arithmetic operation on the two numbers as follows: Child 0 generates S = A+B (addition). Child 1 generates D = A-B (subtraction). Child 2 generates P = A*B (multiplication). Child 3 generates Q = A/B (division). The child program displays the child number, its child PID, and the result of the computation. The sleep time T must be a positive integer no larger than 50. Use a random seed number each time the parent program is run.

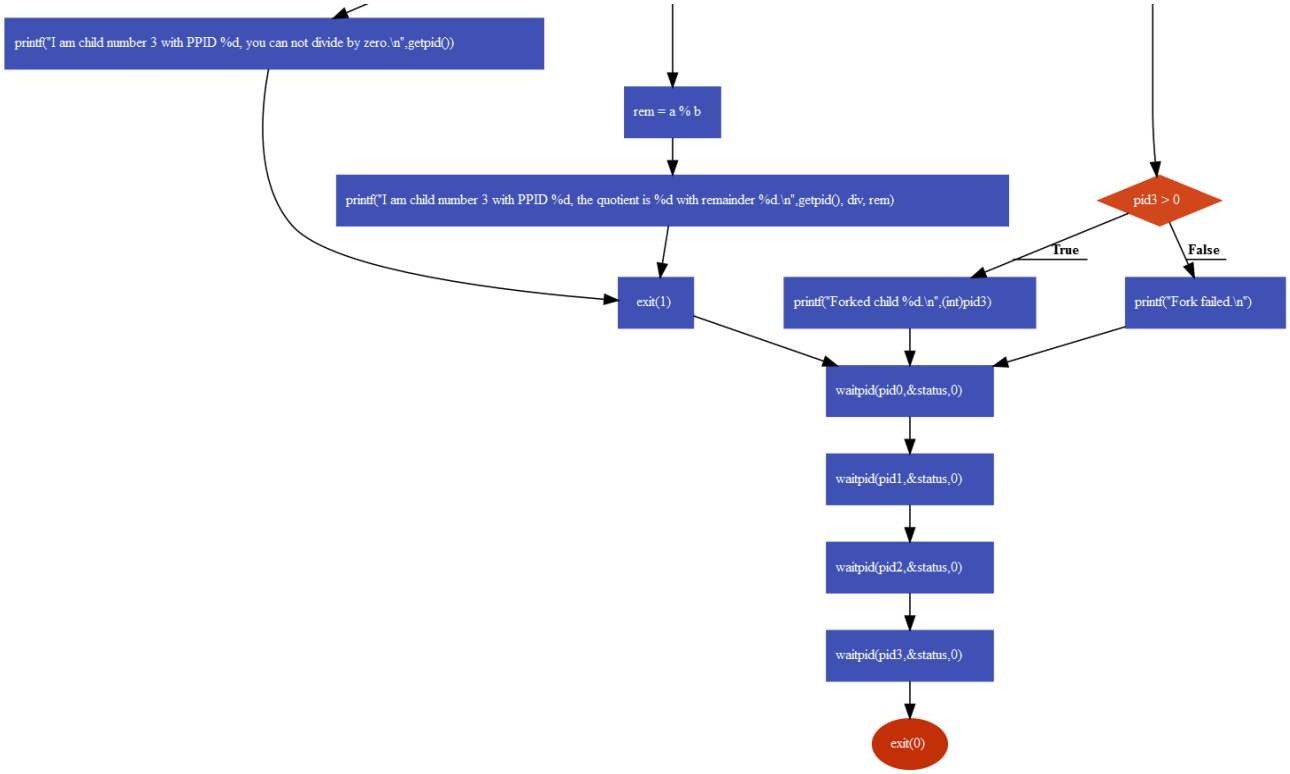
Problem Analysis:

This project is to have the parent process fork 4 children to do the addition, subtraction, multiplication, and division of the 2 numbers given by the user. The parent process will have a maximum of T seconds sleeping time and receives the 2 numbers. The parent process then forks 4 children and sleeps for a random time of mod T. Each children sleeps for a random time of mod T, receives the 2 numbers from the parent process and does their own function. The sleeping time must be a positive number and not larger than 50. The parent prints out the maximum sleeping time and the 2 numbers. Each child prints out the child number, PID, and the result.

Program Flowchart:




True
False
exit(1)
printf("Forked child %d.\n",(int)pid0)
printf("Fork failed.\n")
Creation of child1 process
pid1 = fork()
pid1 == 0
True
seed = (int)(getpid() + pid1) % sleepTime
sleep(seed)
sub = a - b
printf("I am child number 1 with PPID %d, the difference is %d.\n",getpid(), sub)
False
pid1 > 0
True
False
exit(1)
printf("Forked child %d.\n",(int)pid1)
printf("Fork failed.\n")
Creation of child2 process
pid2 = fork()
pid2 == 0
True
seed = (int)(getpid() + pid2) % sleepTime
sleep(seed)
product = a b
printf("I am child number 2 with PPID %d, the product is %d.\n",getpid(), product)
False
pid2 > 0
True
False
exit(1)
printf("Forked child %d.\n",(int)pid2)
printf("Fork failed.\n")
Creation of child3 process
pid3 = fork()
pid3 == 0
True
seed = (int)(getpid() + pid3) % sleepTime
sleep(seed)
b == 0
False
True
False
div = a / b


printf("I am child number 3 with PPID %d, you can not divide by zero.\n",getpid())
rem = a % b
printf("I am child number 3 with PPID %d, the quotient is %d with remainder %d.\n",getpid(), div, rem)
pid3 > 0
True
False
exit(1)
printf("Forked child %d.\n",(int)pid3)
printf("Fork failed.\n")
waitpid(pid0,&status,0)
waitpid(pid1,&status,0)
waitpid(pid2,&status,0)
waitpid(pid3,&status,0)
exit(0)

C Code:

```
# include <stdio.h>
# include <stdlib.h>
# include <sys/types.h>
# include <unistd.h>

int main (int argc, char*argv[])
{
    int a, b, sleepTime, seed, status, sum, sub, product, div, rem;
    pid_t pid0, pid1, pid2, pid3;

    if(argc != 4){
        printf("Need the program name and 3 arguments.\n");
        exit(1);
    }

    sleepTime = atoi(argv[1]);
    a = atoi(argv[2]);
    b = atoi(argv[3]);

    if(sleepTime <= 0 || sleepTime > 50){
        printf("Enter a sleep time that is greater than 0 or 50 and less.\n");
        exit(1);
    }

    printf("I am parent process, the maximum sleep time is %d and the two numbers are %
d and %d.\n",sleepTime,a,b);
    //Creation of child0 process
    pid0 = fork();
    if(pid0 == 0){
        seed = (int)(getpid() + pid0) % sleepTime;
        sleep(seed);
        sum = a + b;
        printf("I am child number 0 with PPID %d, the sum is %d.\n",getpid(), sum);
        exit(1);
    }
    else if(pid0 > 0){
        printf("Forked child %d.\n",(int)pid0);
    }
    else{
        printf("Fork failed.\n");
    }
    //Creation of child1 process
    pid1 = fork();
    if(pid1 == 0){
```

```
        seed = (int)(getpid() + pid1) % sleepTime;
        sleep(seed);
        sub = a - b;
        printf("I am child number 1 with PPID %d, the difference is %d.\n",getpid(), su
b);
        exit(1);
    }
    else if(pid1 > 0){
        printf("Forked child %d.\n",(int)pid1);
    }
    else{
        printf("Fork failed.\n");
    }
    //Creation of child2 process
    pid2 = fork();
    if(pid2 == 0){
        seed = (int)(getpid() + pid2) % sleepTime;
        sleep(seed);
        product = a * b;
        printf("I am child number 2 with PPID %d, the product is %d.\n",getpid(), produ
ct);
        exit(1);
    }
    else if(pid2 > 0){
        printf("Forked child %d.\n",(int)pid2);
    }
    else{
        printf("Fork failed.\n");
    }
    //Creation of child3 process
    pid3 = fork();
    if(pid3 == 0){
        seed = (int)(getpid() + pid3) % sleepTime;
        sleep(seed);
        if(b == 0){
            printf("I am child number 3 with PPID %d, you can not divide by zero.\n",ge
tpid());
        }
        else{
            div = a / b;
            rem = a % b;
        printf("I am child number 3 with PPID %d, the quotient is %d with remainder %d.
\n",getpid(), div, rem);
        }
        exit(1);
```

```
    }
    else if(pid3 > 0){
        printf("Forked child %d.\n",(int)pid3);
    }
    else{
        printf("Fork failed.\n");
    }
    waitpid(pid0,&status,0);
    waitpid(pid1,&status,0);
    waitpid(pid2,&status,0);
    waitpid(pid3,&status,0);
    exit(0);
}
```

Output:

```
ubuntu@ubuntu-xenial:/source/proj2$ ./p2 20 1 2
I am parent process, the maximum sleep time is 20 and the two numbers are 1 and 2.
Forked child 1924.
Forked child 1925.
Forked child 1926.
Forked child 1927.
I am child number 0 with PPID 1924, the sum is 3.
I am child number 1 with PPID 1925, the difference is -1.
I am child number 2 with PPID 1926, the product is 2.
I am child number 3 with PPID 1927, the quotient is 0 with remainder 1.
```