A. Scheduler

Write a program that simulates a scheduler. The program will read the

information about the proccesses from a data file. The scheduler first

reads the number of processes and quamtum size, and then read the arrival

time, burst time, and priority for each process where the lower value

indicates the higher priority. The data file has the following format:

5 2

0 8 4

2 6 2

3 3 1

5 7 3

7 5 5

4 1

1 3 2

2 1 4

4 4 1

6 2 2

6 3

.......

.......

.......

.......

The scheduler then run the processes through FCFS, SJB, SRTN,

preemptive Priority, and RR scheduling algorithms and print the

waiting and turnaround time for each process and the average waiting

and turnaround time for each schedule algrithm. A possible output would

look like:

Round 1: 5 processes, quantum = 2

Waiting |

Time | FCFS | SJF | SRTN | Priority | RR

-----------+------+------+------+----------+------

P1 | 0 | 0 | 3 | 16 | 20

P2 | 6 | 14 | 14 | 3 | 15

P3 | 11 | 5 | 0 | 0 | 9

P4 | 12 | 17 | 17 | 6 | 17

P5 | 17 | 4 | 4 | 17 | 14

-----------+------+------+------+----------+------

Average | 9.2 | 8 | 7.6 | 8.4 | 15

Turnaround |

Time | FCFS | SJF | SRTN | Priority | RR

-----------+------+------+------+----------+------

P1 | 8 | 8 | 11 | 24 | 28

P2 | 12 | 20 | 20 | 9 | 21

P3 | 14 | 8 | 3 | 3 | 12

P4 | 19 | 24 | 24 | 13 | 24

P5 | 22 | 9 | 9 | 22 | 19

-----------+------+------+------+----------+------

Average | 15 | 13.8 | 13.4 | 14.2 | 20.8

Round 2: 5 processes, quantum = 1

.......

.......

.......

.......

The roller coaster problem (60 points)

A theme park contains N people and a roller coaster. The passengers

repeatedly queue up to ride in the car, which can hold maximum C

passengers, where C < N. After the ride the passengers wander

around for a while (up to P minutes) before returning to the roller

coaster for another ride. The car can only go around the track when

it is full, and each journey takes J minutes.

Each person and the roller coaster car must be represented by a separate

process. A person process repeatedly executes the following steps:

o Walk around for a time (random() % P) minutes.

o Wait for the car to arrive, and for space in the car. No passenger may

enter until all previous passengers have left the car.

o Enter the car.

o Wait for the car to finish its journey.

o Exit from the car.

The car process repeatedly executes the following steps:

o Wait for exact C passengers to enter.

o Travel for J minutes.

o Wait for all C passengers to exit.

Notes:

1. The program should terminate after the car has completed T journeys.

2. The program should work for any values of N, C, P, and J. You must

test your program with the values N = 6, C = 3, P = 4, J = 3, T = 20,

but experiment with some other values to check that your program

works correctly.

3. Times can be scaled to reduce execution time; e.g., a minute can be

scaled to 1 seconds in your program.

4. The program should display a suitable message on each of (at least)

the following events:

o A passenger arrives at the roller coaster.

o A passenger enters the car.

o A passenger exits from the car.

Hint: Refer to roller-coster.c.

C. Synchronization Problem (30 points):

Given a set of five processes: A, B, C, D, and E, write a C

program to synchronize the order in which they are executed,

as shown in the following graph:

A ---> B ---> E

\ ^

\ |

+-> C ---> D

That is, process A must finish executing before B or C starts,

process C must finish before D starts, and process B and D must

finish before process E starts.

The code for each process should be fairly simple:

printf("Process A starts...\n");

sleep(rand()%20); /\* Sleep for 0-19 seconds \*/

printf("Process A finishes...\n");

Seed the random number generator with the process id; that is,

srand(getpid());