CSSE 332 – Operating Systems
Rose-Hulman Institute of Technology
Computer Science and Software Engineering Department

Exam 1 – Paper Part

Name:_____ Section:_____ CM:_____

This exam consists of two parts. The first part is to be done on paper without using your computer. The second part of the exam is to be done on your computer. You have the full lab period (a total of three periods) to complete the entire exam.

Instructions: The paper part of the exam is closed book, open notes limited to one double-sided sheet of hand written notes, but **no computer or electronic devices**. Write all of your answers in the spaces provided.

When you complete the paper part of the exam, turn it in to an exam proctor. You may then begin using your computer. **Use of your computer before turning in the paper part of the exam will be considered academic dishonesty.**

To allow sufficient time to work on the computer part of the exam, we suggest using no more than 50 minutes for the paper part.

Please begin by putting your name on the first page and your initials on every page of the exam. We encourage you to skim the entire exam before answering any questions and show all your work to receive partial credit.

	Points available	Your Points
1	10	
2	10	
3	9	
4	9	
5	12	
C	50	
Total	100	

Problem 1 Fill in the Gantt charts, using the given scheduling policy and the table below. Calculate the average turnaround time and average waiting time for each algorithm. If a new process is admitted at the same time that an existing process is preempted, the new process enters the ready queue first. For Multilevel Feedback, there are as many levels as needed and round robin scheduling with the qiven quantum q is used at each level. i is the level of the queue which starts at 0.

Note: processes are always enqueued at the tail of the ready queue and dequeued from the head of the queue.

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

	Time																			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
First-come, first-served (FCFS)	A	A	A	B	B	B	B	B	B											
Round robin, quantum = 4 (RR, q=4)																				
Round robin, quantum = 1 (RR, q=1)																				
Shortest job first (SJF)																				
Shortest remaining time (SRT)																				
Feedback, quantum = 1 (FB, q=1)																				
Feedback, quantum = 2^i (FB, q=2^i)																				

Problem 2 (10 points) Consider the following code.

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

```
3   #include <string.h>
4
5   void processString1(char *string);
6   void processString2(char **string);
7   void processString3(char **string);
8
9   int main(int argc, char *argv[]) {
10    char c;
11    char *string1, *string2, *string3;
12
13    c = 0;
14    string1 = &c;
15
16    processString1(string1);
17    processString2(&string2);
18    processString3(&string3);
19
20    printf("The first string is: %s\n", string1);
21    printf("The second string is: %s\n", string2);
22    printf("The third string is: %s\n", string3);
23
24    return 0;
25  }
26
27  void processString1(char *string) {
28    char lstring[20] = "Hello world!";
29
30    string = lstring;
31
32    return;
33  }
34
35  void processString2(char **string) {
36    char lstring[20] = "Let's try again!";
37
38    *string = lstring;
39
40    return;
41  }
42
43  void processString3(char **string) {
44    char lstring[20] = "Goodbye world!";
45
46    *string = (char *)malloc(sizeof(char)*15);
47    strncpy(*string, lstring, 15);
```

```
48
49      return;
50  }
```

Draw the box and pointer diagram at the follow points:

(a) Just before line 16 executes.

(b) Just before line 32 executes.

(c) Just after line 21 executes.

(d) Just after line 22 executes.

Problem 3 (10 points) Consider the 5-state process model discussed in class.

(a) (3 points) Which state transitions could cause the currently running process to be preempted?

Consider the following sequence of events, which complete in the given order. Assume that the dispatcher runs whenever needed and selects the process at the head of the Ready queue.

- Process P1 is created, admitted, and dispatched.
- Process P2 is created and admitted.
- Process P3 is created and admitted.
- The running process issues I/O request A.
- Process P4 is created and admitted.
- The running process issues I/O request B.
- The running process times out.
- I/O request B completes.
- The running process times out.
- Process P5 is created and admitted.
- The running process exits.

(b) (3 points) Which process is running?

(c) (4 points) Draw the ready queue.

Problem 4 (9 points) Consider the following set of processes, with their length of CPU bursts and arrival times given in milliseconds. Assume that they are scheduled on a uniprocessor system. For the multilevel feedback queue scheduler, use 3 ready queues indexed from 0 to 2. New processes are admitted in queue 0. Note: for preemptive schedulers, newly admitted processes are added to the ready queue before a preempted process when both events happen at the same time.

Process	Burst Time	Arrival Time
P_1	4	0
P_2	8	1
P_3	3	5
P_4	6	7

(a) (6 points) Draw the Gantt charts that illustrate the execution of these processes using the following scheduling algorithms.

Round Robin (quantum = 2)

Shortest Remaining Time

First Come, First Served

(b) (3 points) Compute the average turnaround time for each algorithm over all processes.
Round Robin (quantum = 2)

Shortest Remaining Time

First Come, First Served

Problem 5 (9 points) Consider the following code.

```
1  #include <stdio.h>
2  #include <unistd.h>
3
4  int main(int argc, char* argv[]){
5    int i=0;
6    pid_t forkpid;
7
8    /* Fork a child process */
9    forkpid = fork();
10
11   while (i++ < 4) {
12     if (forkpid < 0){
13       perror("Fork failed. Exiting!");
14       return 1;
15     } else if (forkpid == 0){
16       if ((i%2) == 1) {
17         printf("Child process, my PID = %d.\n", getpid());
18         printf("Child process, fork PID = %d.\n", forkpid);
19       } else {
20         return 0;
21       }
22     }
23     else{
24       printf("Parent process, my PID = %d.\n", getpid());
25       printf("Parent process, fork PID = %d.\n", forkpid);
26     }
27   }
28   return 0;
29 }
```

For this problem assume that `fork()` **always** succeeds. How many child processes are created?

Problem 6 (12 points) **Briefly** answer the following questions.

(a) (3 points) What happens to the other user level threads in a process if one blocks?

(b) (3 points) Compare and contrast a process switch and a mode switch.

(c) (3 points) Why is preemptive scheduling used?

(d) (3 points) What is a cpu-bound processes? What is an I/O-bound processes? Which do you want in your system?