Fill your information below before starting to answer the exam questions.

Name	
ID	
Section	
Instructor	

Instructions

- This is a closed-book exam.
- The exam is six pages, consisting of three questions. Answer all questions.
- Calculators and any smart devices are not allowed in this exam.
- Do not write on the back of any exam pages, write only on the front. If you need additional space, use the extra pages at the end of the exam.

Grading

Question	Points	Result
Q.1 a)	10	
Q.1 b)	10	
Q.1 c)	10	
Q.1 d)	10	
Q.2	30	
Q.3	30	
Total	100	

Q.1 Answer the following questions in the space provided using direct short answers.

a) A process under execution might get suspended/preempted to be resumed later for a variety of reasons. State four of these reasons.

[+2.5 for each of four answers] There are many possible answers, here are some…

1. It exhausts its quantum
2. It voluntarily yields
3. It waits to receive a signal
4. It performs I/O and needs to wait for the response
5. It is suspended by another process through a signal
6. An exception is raised that can be handled
7. Reading from an empty pipe
8. Writing to a full pipe

Note: If a student gave more than four answer, then only their worst four answers were graded.

b) Briefly, explain how the operating system stores the information related to various processes in the system.

[+10] Information about processes is stores in the process control block (PCB). Frequently all the entries are stored in a linked list called the process list.

c) What is the main difference between preemptive and non-preemptive scheduling algorithms? What cautions should be taken with preemptive CPU scheduling?

[+6] Preemptive algorithms allow the running process to be interrupted and replaced by the OS, non-preemptive algorithms do not allow the running process to be forcefully replaced.

[+4] For the cautions there are many possibilities, here are some:

1. When using preemptive algorithms, you need to ensure that data shared between processes remains consistent.
2. With preemptive algorithms, you need to save the process' state when performing a context switch.
3. With preemptive algorithms, you need to worry about context switch overhead.

d) What are the main differences between simulation and implementation/experimentation when studying CPU scheduling algorithms? What is one advantage each has over the other?

[+6] Differences

In simulation, you partially implement an algorithm in a simulator to test and compare whereas in implementation/experimentation you implement the entire algorithm in a real OS in order to test and compare.

[+2] Advantage of simulation over implementation/experimentation

Faster to implement and run tests

[+2] Advantage of implementation/experimentation over simulation

More accurate results using actual workloads.

Q.2 Write a C program that does the following:

- Handles the signal **SIGUSR1** by displaying a message with the format:

  "Parent: Child with PID *pid* exited."

- Creates **10** child processes, where each child process…
  - Ignores the signal **SIGUSR1**
  - Displays a message having the sum of the numbers from **1** up to **max**, a random number between 0 and 10,000 that it generates using the ***rand*** function. The format of the message is:

    "Child: My PID is *pid*. The sum is *sum*"
  - Sends a **SIGUSR1** to the parent
  - Exits
- Waits for the **10** child processes to finish
- Exits

Do not worry about correct #include statements.

Write you answer on this page and the next page.

```
// Handler:
//      2 pts for prototype for a signal handler (sigaction or normal)
//      2 pts for a valid print
//      2 pts for using siginfo and getting the PID
void my_handler(int signum, siginfo_t *si, void *ptr)
{
    printf("Parent: Child with PID %d exited.\n", si->si_pid);
    return;
}

int main(void)
{
    struct sigaction sa;
    int i, k;
    int max,sum;

            // 3 pts for registering a signal properly
    sa.sa_sigaction = my_handler;
    sa.sa_flags = SA_SIGINFO;
    sigaction(SIGUSR1, &sa, NULL);

            // 3 pts for correctly creating 10 children
    for(i = 0; i < 10; i++) {
        if (fork() == 0) {
                                    // 2 pt for ignoring signal
            signal(SIGUSR1, SIG_IGN);

            // 2 pts for correct max generation
            max = rand() % 10000;

            // 2 pts for calculating sum
            sum = 0;
            for(k=1; k < max; k++) {
                sum += k;
            }

            // 3 pts for print with correct arguments
            printf("Child: My PID is %d. Sum is %d\n",
                getpid(), sum);

            // 2 pts for correctly sending signal
            kill(getppid(), SIGUSR1);

            // 1 pts for exit
            exit(0);
        }
    }

    for(i = 0; i < 10; i++) {
        // 3 pts for wait, and it occurring 10 times
        wait(NULL);

    }
}

// 3 pts overall for doing the above in the right places.
```

Q3. Complete the Gantt-chart tracing the execution of the following processes using the given CPU scheduling algorithms. Then complete the table calculating the chosen performance measures of interest. The quantum time in RR is 4 milliseconds.

Processes

PID	Burst Time in milliseconds	Priority
1	9	23
2	4	10
3	6	13
4	3	7
5	8	16

[+5 each algo] Gantt-chart

<table>
<tr><td>Time</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td></tr>
<tr><td>FIFO</td><td colspan="9">P1</td><td colspan="4">P2</td><td colspan="6">P3</td><td colspan="3">P4</td><td colspan="8">P5</td></tr>
<tr><td>PRI</td><td colspan="3">P4</td><td colspan="4">P2</td><td colspan="6">P3</td><td colspan="8">P5</td><td colspan="9">P1</td></tr>
<tr><td>RR</td><td colspan="4">P1</td><td colspan="4">P2</td><td colspan="4">P3</td><td colspan="3">P4</td><td colspan="4">P5</td><td colspan="4">P1</td><td colspan="2">P3</td><td colspan="4">P5</td><td>P1</td></tr>
</table>

[+0.5 Per Time] Performance Measures

PID	Waiting Time			Response Time		
	FIFO	PRI	RR	FIFO	PRI	RR
1	0	21	21	0	21	0
2	9	3	4	9	3	4
3	13	7	19	13	7	8
4	19	0	12	19	0	12
5	22	13	21	22	13	15

Use this page if you need extra space for your answers.