NIA: STUDENT NAME:

Useful functions:

I/O

```
int close(int fd)
int open(const char *path, int oflag)
int write(int fd, void *buf, int nbyte)
int read(int fd, void *buf, int nbyte)
int lseek(int fd, int offset, int whence)
      where whence == SEEK_SET: beginning of the file offset
            whence == SEEK_CRT: current file pointer offset
            whence == SEEK_END: end of file offset
```

Process Management

```
int wait(int *status)
pid_t fork(void)
void exit(int status);
```

I. **(20 points)** Given the following set of processes:

Process	Burst Time (ms)	Arrival time	Priority
*P*1	12	0	1
*P*2	4	2	3
*P*3	2	5	1
*P*4	10	8	3
P5	6	2	2

a. Show the execution order of these processes as a Gantt chart for the following scheduling policies: FCFS, ***non-premptive SJF, preemptive SJF, Round-Robin (quantum 4) and preemptive priority.***

b. Calculate the ***average waiting time*** and ***average turnaround time*** for all these policies.

Solution:

FCFS

P1	P2	P5	P3	P4
0	12	16	22	24

Average waiting time: (0+10+17+16+14)/5 = 57/5

Average turnaround time: (34 + 57)/5

Non-preemptive SJF

P1	P3	P2	P5	P4
0	12	14	18	24

Average waiting time: (0+12+7+16+16)/5 = 51/5
Average turnaround time: (34 + 51)/5
Preemptive SJF

P1	P2	P3	P5	P1	P4
0	2	6	8	14	24

Average waiting time: (12+0+1+16+6)/5 = 35/5
Average turnaround time: (34 + 35)/5

RR

P1	P2	P5	P3	P4	P1	P5	P4	P1	P4
0	4	8	12	14	18	22	24	28	32

Average waiting time: (20+2+7+16+16)/5 = 61/5
Average turnaround time: (34 + 61)/5

Preemptive priority

P1	P3	P5	P2	P4
0	12	14	20	24

Average waiting time: (0+18+7+16+12)/5 = 53/5
Average turnaround time: (34 + 53)/5

NIA: STUDENT NAME:

II. **(30 points)** Develop a small program in C, which uses system calls for providing the following functionality. A father reads from the keyboard N integers and writes them to a file. Subsequently, it creates N/1024 children. Each child reads 1024 integers from the file, calculates the sum and returns the sum to the father. The father retrieves the partial sums from the children and calculates their total. Finally, it prints the result and terminates.

Points:

6 Read keyboard + Write file
4 Read file
5 Create children
10 Correct father/child separation
5 Wait

Solution:

```
fd = creat(fname,0666);
for (i=0;i<N;i++) {
        read(0,&nr, sizeof(int));
        write(fd,&nr, sizeof(int));
}
total=0;
for (i=0;i<N/1024;i++) {
        if (fork()==0) {
                sum = 0;
                lseek(fd,i*sizeof(int), SEEK_SET);
                for (j=0;j<1024;j++) {
                        read(fd, &num,sizeof(int));
                        sum+=num;
                }
                exit(num);
        }
        else {
        wait(&status);
        total+=status;
        }
}
printf("total=%d", total;
```

NIA: STUDENT NAME:

III. **(20 points)** A UNIX program is using file system calls for reading writing to/from a file. Indicate the contents of the file and the value of the file pointer after each of the operations labeled with 1, 2, 3, 4, and 5.

```
char buf[11]="0123456789";
char buf2[4];
int fd=creat("filename",0666);
```

1: write(fd,buf,10);

File pointer value: 10

File offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Content	0	1	2	3	4	5	6	7	8	9					

2: lseek(fd, 2, SEEK_SET);

File pointer value: 2

File offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Content	0	1	2	3	4	5	6	7	8	9					

3: read(fd, buf2, 4);

File pointer value: 6

File offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Content	0	1	2	3	4	5	6	7	8	9					

4: lseek(fd, 2, SEEK_CUR);

File pointer value: 8

File offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Content	0	1	2	3	4	5	6	7	8	9					

5: write(fd, buf2, 4);

File pointer value: 12

File offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Content	0	1	2	3	4	5	6	7	2	3	4	5			

NIA: STUDENT NAME:

IV. **(10 points)** Quiz questions (***NOTE: A correct answer scores 1point, a wrong answer subtracts 0.25 points, an unanswered question scores 0 points***)

	1	2	3	4	5	6	7	8	9	10
Answer										

1. Which one is NOT a system call parameter passing method:
a) by stack b) by registers c) by pipe d) by memory block.
2. A file is not:
   a) a storage abstraction b) a collection of blocks stored on disk c) a directory d) a cache
3. In the blocked state you can find:
a) the processes waiting for I/O b) the process which is running c)the processes waiting for the processor d) none of the above
4. Which of the following scheduling is based on swapping
   a) short-term b) medium-term c) long-term d) none.
5. Saving the CPU state of the old process and loading a new process CPU state is called:
   a) Process Blocking b) Context Switch c) Time Sharing d) None of the above
6. Which affirmation is not true about the differences between a thread and a process?
   a) a process is heavyweight, while a thread is lightweight b) A thread is a part of a process c) A process must have at least one thread d) The threads and processes share the same stack.
7. Which one is not a thread benefit?
   a) Threads share resources inside each process b) Scalability c) Responsiveness d) Protection
8. Context switching between threads is more expensive than context switching between processes.
   a) True b) False
9. Which of the following components of program state are shared across threads in a multithreaded process?
   a) Register values b) Heap memory c) Stack memory d) Program counter
10. Without kernel level support, a blocking user-level thread will block all the other threads in the same process.
    a) True b) False

V. **(20 points)** Give definitions of the following concepts in maximum 3 lines:
1. System mode
2. Microkernel
3. Kernel threads
4. Preemptive scheduling
5. Time sharing

6. Process control block

Solution: see Silberschatz book