CS 304 : Mid Semester Examination	Marks obtained ↓
Date: 21.02.2022, Total questions: **10** Total Marks: **37**	
Name: Roll No:	Time: 2 hrs

Question:	1	2	3	4	5	6	7	8	9	10	Total
Points:	2	3	3	5	4	5	3	4	4	4	37
Score:											

Instructions:

1. **Write down your name and roll no. at the top right side of every page of the answer sheet.**
2. **Please make any assumptions that you deem to be reasonable.**
3. **You are not allowed to use any electronic gadgets including calculators and mobile phones.**

[2] 1. Five jobs are waiting to be run. Their expected run times are 9, 6, 3, 5, and 'X'. In what order should they be run to minimize average response time (of course your answer will depend on X). Explain the different possible cases based on the value of 'X'.

[3] 2. Consider the following preemptive–priority-scheduling algorithm based on dynamically changing priorities. Larger priority numbers imply higher priority. When a process is waiting for the processor (in the ready queue, but not running), its priority changes at a rate A [i.e P(t) = P0 +A * (t - t0) where t0 is the time at which the process joins the ready queue] ; when it is running, its priority changes at a rate B. All processes are given a priority of zero when they enter the ready queue. The parameters A and B can be set to give many different scheduling algorithms.

(A) What is the algorithm that results from $B > A > 0$? Explain how

(B) What is the algorithm that results from $A < B < 0$? Explain how

[3] 3. Assume the OS schedules a workload containing three jobs with the following characteristics:

Process	Arrival Time	Burst Time
A	0	10
B	5	8
C	12	2

(A) Which scheduler minimizes the completion time of the entire workload? Explain why?

(B) Given a STCF scheduler, what is the response time for job B?

(C) Given a STCF scheduler, what is the response time for job C?

[5] 4. (A) Suppose there are n instances of a resource in a system, each time one instance of resource can be acquired by a process and utilized exclusively. Explain how semaphores can be used to ensure the correct usage of the resource.

(B) What is the difference between deadlock prevention and deadlock avoidance? What category does Banker's algorithm fall into and why?

(C) Suppose a thread is running in a critical section of code, meaning that it has acquired all the locks through proper arbitration. Can it get context switched? Why or why not?

[4] 5. Here is source code for a program, called **val_inc.c**:

```c
int value = 0;
int main(int argc, char*argv[]) {
   while (1) {
      printf("%d", value);
      value++;
   }
return 0;
}
```

While **val_inc.c** is running, another program, **val_res.c**, is run once as a separate process.
Here is the source code of **val_res.c**:

```c
int value;
int main(int argc, char*argv[]) {
     value = 0;
     return 0;
}
```

Which of the following below are possible outputs of the **val_inc** process? Explain your answer for each case.

(A) 012345678

(B) 012345670123

(C) 01234567891011

(D) 123456789

5 6. (A) Look at the C program below. You can assume there won't be errors. include's have been left out. *[FYI: waitpid(pid, status, 0) waits for process pid to finish.]* Write its output and explain your answer.

```c
char array[] = "qwertyuiopasdfghjklzxcvbnm";
int len = sizeof(array);// 26 initially
void mystery(){
     char first = array[0];
     int status, i;
     int j = 0;
     int pid = fork();
     if(pid == 0){
         for(i= 1; i< len; i++)
            if(array[i] < first)
               array[j++] = array[i];
      }
      else{
         waitpid(pid, &status, 0);// wait for pid
         write(1, &first, 1); // prints first
         for(i= 1; i< len; i++)
            if(array [i] > first)
               array[j++] = array[i];
      }
      len = j;
      if(len> 0)
           mystery();
      }
int main(){
    mystery();
    return 0;
}
```

(B) In the code given below, a semaphore is used to wait for the child threads to complete. The parent thread creates three children. What should the value, semaphore s be initialized to?

```
1  sem_t s;
2  void child(void *arg) {
3   // do some stuff, then signal completion
4  sem_post(&s)
5  }
6
7  int main(int argc, char *argv[]) {
8      thread_t p1, p2, p3;
9      sem_init(XXX);
10     thread_create(&p1, child);
11     thread_create(&p2, child);
12     thread_create(&p3, child);
13     sem_wait(&s);
14 }
```

3 7. Assume the following code is compiled and run on a linux machine. Also assume that any irrelevant details have been omitted and the pthread_* routines does not throw any error.

```
1  volatile int balance = 0;
2  void *mythread(void *arg){
3       int result = 0;
4       result = result + 375;
5       balance = balance + 375;
6       printf("Result is %d\n", result);
7       printf("Balance is %d\n", balance);
8       return NULL;
9  }
10 int main(int argc, char *argv[]){
11     pthread_t p1, p2;
12     pthread_create(&p1, NULL, mythread, "A");
13     pthread_create(&p2, NULL, mythread, "B");
14     pthread_join(p1, NULL);
15     pthread_join(p2, NULL);
16     printf("Final Balance is %d\n", balance);
17 }
```

(A) How many total threads are part of this process?

(B) When thread p1 prints "Result is %d", what value of result will be printed? Elaborate your answer.

(C) When thread p1 prints "Balance is %d", what value of balance will be printed? Elaborate your answer.

(D) When "Final Balance is %d" is printed, what value of balance will be printed? Elaborate your answer.

4 8. Recall the five Dining Philosophers (and five chopsticks) sitting around a circular table:

(A) Assume L philosophers are left-handed and pick up their left chopstick first, while the remaining R philosophers are right-handed and pick up their right chopstick first. Fill the below table indicating yes, no, or depends. Explain your answer for each case

L	R	Deadlock Possible?
5	0	
4	1	
3	2	

(B) Suppose now, the five philosophers instead decide to place N chopsticks in a heap in the middle of the table. Philosophers can now pick up one chopstick at a time, but still need two to eat. Say if a deadlock is possible (yes, no, or depends). Explain your answer for each case.

N	Deadlock Possible?
> 5	
5	
< 5	

(C) Consider a table with five three-handed philosophers and a pile of N chopsticks in the middle of the table. Each philosopher needs 3 chopsticks to eat. What is the smallest N such that a deadlock is impossible?

4 9. Consider a barbershop consisting of a room with N chairs. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy, but chairs are available, then the customer sits in one of the free chairs and awaits his turn. The barber moves onto the next waiting seated customer after he finishes one haircut. If there are no customers to be served, the barber goes to sleep. If the barber is asleep when a customer arrives, the customer wakes up the barber to give him a haircut. A waiting customer vacates his chair after his hair cut completes. Your goal is to write the pseudocode for the customer and barber threads below with suitable synchronization. You must use only semaphores to solve this problem. Use the standard notation of invoking up/down (sem_post/sem_wait) functions on a semaphore variable.
The following variables (3 semaphores and a count) are provided to you for your solution. You must use these variables and declare any additional variables if required.

```
semaphore mutex = 1, customers = 0, barber = 0;
int waiting_count = 0;
```

Some functions to invoke in your customer and barber threads are:
A customer who finds the waiting room full should call the function `leave()` to exit the shop permanently. This function does not return. A customer should invoke the function `getHairCut()` in order to get his haircut. This function returns when the haircut completes. The barber thread should call `cutHair()` to give a haircut. When the barber invokes this function, there should be exactly one customer invoking `getHairCut()` concurrently.

4 10. Consider the following snapshot of a system in which four resources A, B, C and D are available.

	Allocation				*Max*				*Need*				*Available*			
	A	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*
P_0	2	0	1	1	3	2	1	1					6	4	4	2
P_1	1	1	0	0	1	2	0	2								
P_2	1	0	1	0	3	2	1	0								
P_3	0	1	0	1	2	1	0	1								

Answer the following using Banker's algorithm.

(A) Compute what each process might still request and fill this in under the column "Need"

(B) Is the system in a safe state? Why or why not?

(C) Is the system deadlocked? Why or why not?

(D) If a request from P3 arrives for (2,1,0,0), can the request be granted immediately ?