National University of Computer & Emerging Sciences, Karachi
Spring 2021 CS-Department
Final Examination
08th June 2021, 12:30 pm – 02:00 pm Spring 2021

Course Code: CL 220	**Course Name: Operating Systems Lab**
Instructor Name / Names: Tania Iram, Anaum Hamid, Safia, Rabia, Ali Fatmi	
Student Roll No: 19K-0207	**Section No:** L
PaperBBBBBBBBBBBBBBBBBBBB	

Instructions:

- Return the question paper.
- Read each question completely before attempting it. There are **5 questions and 4 pages**
- In case of any ambiguity, you may make assumption. But your assumption should not contradict any statement in the question paper.

Max Marks: 50 points

Time: 90 minutes.

Q1. Signal Handling **Marks 5**

Complete the program that implements Alarm function. Program keep printing "Alarm is on" after each second five times, then it turns off for five seconds. When alarm is off it keeps printing "Alarm is off". Implement an interrupt call which snoozes alarm and prints "Alarm is snoozed". Implement a stop call that stops the alarm and prints "Alarm is stopped" & terminates the program.

```
#include<stdio.h>
#include<unistd.h>
#include<wait.h>
#include<signal.h>
#include<errno.h>
#include <stdlib.h>
int check=0;

void wakeup() {

}
void snooze( int signum) {

}
void stop( int signum) {

}

int main() {
signal(SIGALRM,wakeup);
signal(SIGINT, snooze);
signal(SIGTSTP, stop);
while(1){
        alarm(5);
        pause();
        printf("Alaram is of\n");

}

return 0;

}
```

Q2. Shell scripting and Process Management and Communication **Marks 10**

a) Write a program 'priority.sh' to open firefox with -4 priority

b) write a c program to send the name of the shell script in part a from parent process to child process using pipes. Child will then execute that shell script and exit.

C program	priority.sh

Q3: Multithreaded Programming using Pthreads **Marks 10**

Complete the code of a simple multithreaded program in C that create multiple threads by default attributes on Linux environment and each thread will print the table till 10 with its Id number. Perform necessary error checking where needed. In given Code Grey area. main has 1 and in thread has more than 2 lines missing.

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#define N 4
void *child_thread(void *arg)
{




}
int main()
{       pthread_t my_thread[N];
      long id;
      for(id = 1; id <= N; id++) {
            int value =
            if(value != 0) {
                  printf("Error: pthread_create() failed\n");
                  exit(EXIT_FAILURE);              }
      }           pthread_exit(NULL);
}
```

Q4: Multithreaded Programming using OpenMp **Marks 15**

In stock market the shares of companies go up and down during a day. Write an OpenMp program for stock market shares update according to the given requirement:

- You need to initialize two arrays with random values.
- Array 'a[8]' will hold the opening price of stock for 8 companies.
- Array d[8] will be initialized with random values between one to ten. (use rand() function).
- Four openmp threads will update the values of two companies ,once only, in each iteration after checking the contents of d[i] (criteria is increase the share value by d[i] value if value greater than 5 otherwise decrease by the value of d[i]).
- Display which thread update the value of which company
- Display the closing price of shares of each

Q5: Process Synchronization **Marks 10**

Complete the code for a synchronize X-ray treatment mechanism in the hospital using semaphore where there are 20 clients and 2 attendants, each needs to wait to get his appointment token/number. During waiting and treatment, patients feel sleepy so they sleep, weighting area takes 5 seconds sleep, and the attendant will usually take 20 sec for treatment, so patient need 20 seconds sleep. Main function is given Write the handler function code only.

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<semaphore.h>
#include<string.h>
#include<unistd.h>
sem_t appointment, waiting;

void handler(void* ptr) {

}
int main() {
int i,a[10];
pthread_t patient[20];
sem_init(&appointment,0,1);
sem_init(&waiting,0,2);
for(i=0;i<20;i++) {
a[i]=i;
pthread_create(&patient[i],0, (void*)handler, (void*)&a[i]);
}
for(i=0;i<20;i++) {
pthread_join(patient[i],NULL);

}
printf("Done");
//sem_destory(&appointment);
//sem_destory(&waiting);
return 0;
}
```