0200

National University of Computer and Emerging Sciences
Islamabad Campus

Operating Systems (CS2006)

Sessional-II Exam

Total Time (Hrs): 1
Total Marks: 55
Total Questions: 4

Course Instructor(s):
Dr. Ali Zeeshan,Ms.Sidra Khalid, Ms.Sana Razzaq,
Ms. Rabail Zahid
Section(s): (CS- A,B,C,D,E,F,G)

Date: Apr 8, 2025

______________ ______________
Roll No **Course Section**

Do not write below this line.

Attempt all the questions.

All Question are of type Subjective and required to be attempted on the Answer Sheet.

[CLO 1: Understand, design and implement solutions employing concepts of Processes and Threads.]

Q1: Threads

[15 marks]

Part A

[Marks 5]

Write a C/C++ program that performs the following tasks:

1. Create **five threads** using the pthread library.
2. Each thread should compute the square of its thread number and return the result.
3. The main thread should collect the results and print the square computed by each thread.

Ensure proper error handling during thread creation and joining.

Part B

[Marks 10]

Provide output of given program

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_THREADS 4
void *thread_function(void *arg) {
    int thread_id = *(int *)arg;
    printf("Thread %d: Starting work...\n", thread_id);
    sleep(2);
    printf("Thread %d: Finished work.\n", thread_id);
    pthread_exit((void *)(long)thread_id);
}
int main() {
   pthread_t threads[NUM_THREADS];
   pthread_attr_t attr;
   int thread_ids[NUM_THREADS];
   void *status;
   int rc;
   pthread_attr_init(&attr);
for (int i = 0; i < NUM_THREADS; i++) {
     thread_ids[i] = i + 1;
     if (i % 2 != 0) {
       pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_DETACHED);
       printf("Creating detached thread %d\n", thread_ids[i]);
      } else {
       pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);
       printf("Creating joinable thread %d\n", thread_ids[i]);
      }

rc=pthread_create(&threads[i],&attr,thread_function,&thread_ids[i]);
  if (rc) {
      fprintf(stderr, "Error: Unable to create thread %d\n", rc);
       exit(EXIT_FAILURE);
      }
}
  pthread_attr_destroy(&attr);
  for (int i = 0; i < NUM_THREADS; i += 2) {
        rc = pthread_join(threads[i], &status);
        if (rc) {
          fprintf(stderr, "Error: Unable to join thread %d\n", rc);
          exit(EXIT_FAILURE);
        }
   printf("Main: Joined thread %d with result %ld\n",
   thread_ids[i], (long)status);
   }
  printf("Main: Program completed.\n");
  pthread_exit(NULL);
}
```

[CLO 2: Compare and contrast the commonly used for scheduling tasks in operating systems and analyze different synchronization issues and implement synchronization mechanisms like Semaphores, Monitors, etc.]

Q2: Software Solution **[15 marks]**

Part A **[8 marks]**

- Process P_i

```
do {
   flag [ i ]:= true;
   turn = j;
   while (flag [ j ] and turn = j);

      critical section

   flag [ i ] = false;
      remainder section
} while (1);
```

- Process P_j

```
do {
   flag [ j ]:= true;
   turn = i;
   while (flag [ i ] and turn = i);

      critical section

   flag [ j ] = false;
      remainder section
} while (1);
```

Consider Peterson's solution for two processes (P0 and P1). Suppose the following sequence of events occurs:

1. P0 starts executing and sets flag[0] = true;.
2. P0 gets preempted before setting turn = 1;.
3. P1 starts executing, sets flag[1] = true;, and then sets turn = 0;.
4. Both processes reach the while loop condition at the same time.

Questions:

a. Which process will enter the critical section first? Why or why not[2]

b. What will be value of turn variable.[1]

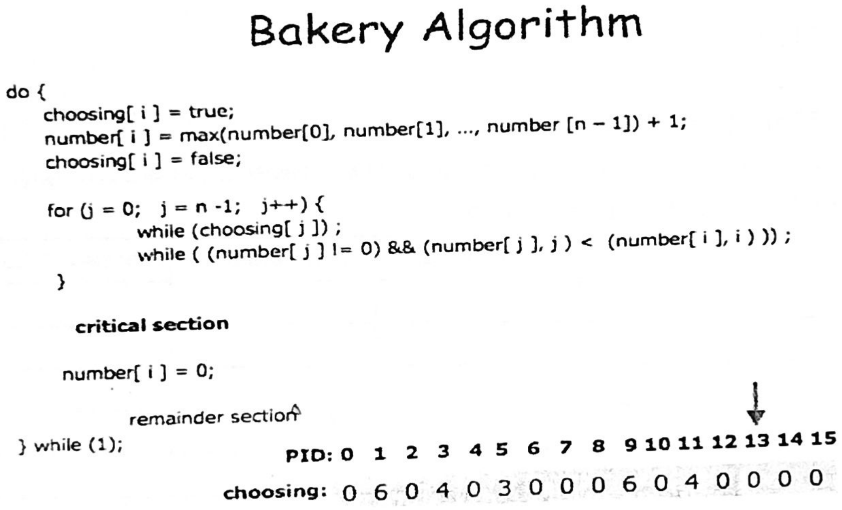
c. If CPU pause P0 to handle an external event (e.g., I/O operation) for a long time, will P1 be able to enter the critical section? Why or why not?[1]

d. Does this scenario violate mutual exclusion? Why or why not?[2]

e. How does the use of the turn variable help resolve conflicts in such cases?[2]

[7 marks]

Part B



Bakery Algorithm

```
do {
    choosing[ i ] = true;
    number[ i ] = max(number[0], number[1], ..., number [n – 1]) + 1;
    choosing[ i ] = false;

    for (j = 0;  j = n -1;  j++) {
              while (choosing[ j ]) ;
              while ( (number[ j ] != 0) && (number[ j ], j ) <  (number[ i ], i ) )) ;
     }

      critical section

     number[ i ] = 0;

              remainder section
} while (1);
```

PID:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
choosing:	0	6	0	4	0	3	0	0	0	6	0	4	0	0	0	0

a. (True or False) The above ticket numbers are not correct since two processes received ticket numbers 6. [1 Mark]

b. (True or False) This algorithm can be used for 2-process mutual-exclusion problem. [1 Mark]

c. (True or False) The lines "choosing[i] = true;" and "choosing[i]=false;" provide mutual exclusion access to line "number[i]=max(number(0) ....)+1;". [1 Mark]

d. In the array of "choosing" shown at the bottom of the figure, add a proper ticket number for the PID=13.[1]

e. Specify the order that different processes can enter to their critical sections. Indicate each process with its PID.[3 Marks]

[CLO 2: Compare and contrast the commonly used for scheduling of tasks in operating systems and analyze different synchronization issues and implement synchronization mechanisms like Semaphores, Monitors, etc.]

Q3: Synchronization-Hardware Solution **[10 marks]**

Three threads (**T1, T2, T3**) use the **`TestAndSet`** function to implement a spinlock. The initial value of the boolean lock is **FALSE**. The threads execute operations in the following order:

1. T1 calls `TestAndSet(&lock)`.
2. T2 calls `TestAndSet(&lock)`.
3. T1 exits the critical section.
4. T3 calls `TestAndSet(&lock)`.
5. T3 exits the critical section.
6. T2 calls `TestAndSet(&lock)` again.

bool r = *target
*target = true
return r;

Copy the following table on the Answer sheet and fill this table on the Answer sheet accordingly. Zero Marks will be awarded if table is filled on question paper.

Step	Thread	Operation	Return Value	Lock Value	Thread in CS
1					
2					
3					
4					
5					
6					

Note: write "N/A" for releases in return value, "None" if no thread is in CS.
Assume atomic execution of `TestAndSet` and that a thread enters the CS only if `TestAndSet` returns FALSE. Ignore spinning behavior (only track explicit calls listed).

[CLO 2: Compare and contrast the commonly used for scheduling of tasks in operating systems and analyze different synchronization issues and implement synchronization mechanisms like Semaphores, Monitors, etc.]

Q4: Consider the following set of processes, with the arrival time and CPU burst time given in milliseconds: **[15 marks]**

Process	Arrival Time(ms)	Burst Time(ms)
P1	0	30
P2	19	15
P3	42	24
P4	27	10
P5	3	59

a. Draw three Gantt charts to show the execution of these processes using the scheduling algorithms listed below:

   i. Shortest-Job-First [3 marks]
   ii. Shortest-Remaining-Time-First [3 marks]
   iii. Round-Robin (quantum = 8) [3 marks]

b. What is the average turn around for each scheduling algorithm in part a? [1+1+1]

c. What is the average waiting time for each scheduling algorithm in part a? [1+1+1]

NOTE -We will be evaluating your final answer for part b and c, but make sure to mention values of turnaround time and waiting time for each process. Final answer without intermediate calculations will be marked Zero.

GoodLuck