Q1.1 *

Suppose a banking system has two methods: deposit() and withdraw(), used by clients to deposit and withdraw a certain amount passed as a parameter from their account, respectively. Suppose the algorithms for withdraw() and deposit() are as shown below. A Boolean flag is used for synchronization initialized to false.

```
deposit(amount)
{
      while (flag) ;
      flag = true;
      balance+= amount;
      flag = false;
}
```

```
withdraw(amount)
{
        while (flag) ;
        flag = true;
        balance -=amount;
        flag = false;
}
```

What is (are) the name(s) of the shared variables in this example.

Mark only one oval.

- amount
- balance
- flag
- balance and amount
- balance and flag
- balance, amount and flag

5. Q1.2 *

Suppose a banking system has two methods: deposit() and withdraw(), used by clients to deposit and withdraw a certain amount passed as a parameter from their account, respectively. Suppose the algorithms for withdraw() and deposit() are as shown below. A Boolean flag is used for synchronization initialized to false.

```
deposit(amount)
{
      while (flag) ;
      flag = true;
      balance+= amount;
      flag = false;
}
```

```
withdraw(amount)
{
        while (flag) ;
        flag = true;
        balance -=amount;
        flag = false;
}
```

Does the proposed solution satisfy the mutual exclusion requirement?

Mark only one oval.

- Solution satisfies mutual exclusion requirement
- Solution violates mutual exclusion requirement

6. Q1.3 *

Suppose a banking system has two methods: deposit() and withdraw(), used by clients to deposit and withdraw a certain amount passed as a parameter from their account, respectively. Suppose the algorithms for withdraw() and deposit() are as shown below. A Boolean flag is used for synchronization initialized to false.

```
deposit(amount)
{
      while (flag) ;
      flag = true;
      balance+= amount;
      flag = false;
}
```

```
withdraw(amount)
{
        while (flag) ;
        flag = true;
        balance -=amount;
        flag = false;
}
```

Now consider a husband and wife sharing an account. The initial balance in the account is SR 500. Concurrently, the husband calls deposit(100) and wife calls withdraw(50). What would be the value of balance after both operations complete concurrently.

Mark only one oval.

- 450
- 600
- 550
- Any of the above
- None of the above

Q1.4 *

Suppose a different solution is used instead, that utilizes two flags initialized to false instead as follows. Does the proposed solution satisfy the progress requirement?

```
deposit(amount)
{
     while (flag[1]) ;
     flag[0] = true;
     balance+= amount;
     flag[0] = false;
}
```

```
withdraw(amount)
{
        while (flag[0]) ;
        flag[1] = true;
        balance -=amount;
        flag[1] = false;
}
```

Mark only one oval.

- Solution satisfies the progress requirement
- Solution violates the progress requirement

Q1.5 *

Suppose that a semaphore `sync` is used instead to coordinate the two methods `deposit()` and `withdraw()`.

What would be a proper line(s) to place the statement `wait(sync);` ?

deposit(amount)	withdraw(amount)
{	{
_____(D1)	_____(W1)
balance += amount; (D2)	balance -= amount; (W2)
_____(D3)	_____(W3)
}	}

Mark only one oval.

- Line W1
- Line W3
- Line D1 and Line W3
- Line W1 and Line D3
- Line D1 and Line W1
- Line D3 and Line W3

Q1.6 *

Suppose that a semaphore `sync` is used instead to coordinate the two methods `deposit()` and `withdraw()`.

What would be a proper line(s) to place the statement `wait(sync);` ?

deposit(amount)	withdraw(amount)
{	{
_____(D1)	_____(W1)
balance += amount; (D2)	balance -= amount; (W2)
_____(D3)	_____(W3)
}	}

What would be the initial value of the semaphore `sync`?

Mark only one oval.

- -1
- 0
- 1
- 2

Q2.1 *

Suppose a buffer of 10 slots is shared between 2 producers and 3 consumer processes. A semaphore ***full*** is utilized to track the number of full slots.
Assume the buffer is initially empty. What would be the initial value of semaphore ***full***?

Mark only one oval.

- 0
- 1
- 2
- 10

Q2.2 *

Suppose a buffer of 10 slots is shared between 2 producers and 3 consumer processes. A semaphore ***full*** is utilized to track the number of full slots.

Assume the producers have produced two items into the buffer. What would be the value of semaphore ***full***?

Mark only one oval.

- 0
- 1
- 2
- 10

Q3.1 *

Assume the following pseudocodes synchronize the execution of threads T1, T2 and T3 by using "**no Busy Waiting**" implementation utilizing two semaphores: ***S*** and ***Q***, in order to print the output in the following manner: **2 1 .**

```
int x = 0;      semaphore S;
int y = 0;      semaphore Q;
```

T1	T2	T3
While(x<2){ x++; } signal(S);	------- // **L1** print (x); ------- // **L2** print (y);	y++; signal(Q);

What would be the initial values of semaphore ***S*** and ***Q***:

Mark only one oval.

- S = 0 , Q = 0
- S = 1 , Q = 1
- S = 0 , Q = 1
- S = 1 , Q = 0

Q3.2 *

Assume the following pseudocodes synchronize the execution of threads T1, T2 and T3 by using "**no Busy Waiting**" implementation utilizing two semaphores: ***S*** and ***Q***, in order to prin the output in the following manner: **2 1 .**

```
int x = 0;      semaphore S;
int y = 0;      semaphore Q;
```

T1	T2	T3
While(x<2){ x++; } signal(S);	------- // **L1** print (x); ------- // **L2** print (y);	y++; signal(Q);

What would be the appropriate operation and utilized semaphore in line **L1**:

Mark only one oval.

- wait(S);
- wait(Q);
- signal(S);
- signal(Q);

4. Q3.3 *

Assume the following pseudocodes synchronize the execution of threads T1, T2 and T3 by using "**no Busy Waiting**" implementation utilizing two semaphores: **S** and **Q**, in order to print the output in the following manner: **2 1**.

```
int x = 0;        semaphore S;
int y = 0;        semaphore Q;
```

T1	T2	T3
`While(x<2){ x++; }` `signal(S);`	`------- // L1` `print (x);` `------- // L2` `print (y);`	`y++;` `signal(Q);`

What would be the appropriate operation and semaphore in line **L2**:

Mark only one oval.

- wait(S);
- wait(Q);
- signal(S);
- signal(Q);

Question 4

The dining philosophers' problem is a classical synchronization problem. Answer the following related questions.

Q4.1 *

How is deadlock possible in the dining philosophers' problem with five philosophers?

Mark only one oval.

- More than one philosopher attempts to eat at the same time.
- Every philosopher holds the left chopstick and waits indefinitely for the right chopstick.
- At most four philosophers are allowed to sit on the table at any time.
- An asymmetric order of picking chopsticks is applied among even-positioned and odd-positioned philosophers.

Q4.2 *

In the dining philosophers' problem, a philosopher may access the shared rice bowl available at the center of the table using:

Mark only one oval.

- the chopstick to his left only.
- the chopstick to his right only.
- either the chopstick to his left or the chopstick to his right.
- both the chopstick to his left and the chopstick to his right.

Q5.1 *

Suppose a system of 10 reader and 5 writer processes are shared a file implemented using first reader-writer problem. What is the maximum number of reader processes that can access to the available file simultaneously?

Mark only one oval.

- 0
- 1
- 2
- 10

Q5.2 *

Suppose a system of 10 reader and 5 writer processes are shared a file implemented using first reader-writer problem. What is the maximum number of writer processes that can access to the available file simultaneously?

Mark only one oval.

- 0
- 1
- 5
- 10