JOY KARMOKAR
18-39263-3 (OS-D)

1]

```
#!/bin/bash
echo " users:"
read u
for ((i=1; i<=$u; i++))
do
echo " person $i"
for subject in "OS" "COA" "CN" "CAN" "HCI"
do
echo "mark of $subject :"
read n
if [ $n -ge 90] && [ $n -le 100]
then
   echo " Grade of $ subject = A+"
elif [ $n -ge 85] && [$n - lt 90]
 then
   echo " Grade of $ subject = A "
elif [$n - ge 80] && [ $n - lt 85]
  then
    echo " Grade of $ subject =B+"
elif [ $n - ge 75] && [ $n - lt 80]
  then
    echo " Grade of $ subject = B"
```

```
elif [$n -ge 70] && [$n -lt 75]
 then
   echo "grade of $subject = C+"
 elif [ $n -ge 65] && [$n -lt 70]
  then
   echo "grade of $ subject = C"
 elif [ $n -ge 60] && [$ n -lt 65]
   then
    echo " grade of $ subject = D+"
 elif [ $n -ge 50] && [ $n -lt 60]
    then
     echo " grade of $ subject =D"
 elif [ $n -lt 50]
  then
   echo " grade of $subject = F"
 else
   echo "  "
 fi
done
 done
```

3)

```
#!/bin/bash
echo "menu"
echo "1. print date and time"
echo "2. List all files in the current derectory"
echo "choose 1 or 2"
read option
if [$option == 1]
 then
     echo 'date'
elif [$option == 2]
  then
     echo '-ls'
  else
     echo "other"
fi
```

3|

ascending and descending order.

```
Cat > text.txt

Eng
Math
DLC
TOC
CN
```

ASCending order,

```
sort text.txt

CN
DLC
Eng
Math
TOC
```

Descending order,

```
sort -r text.txt

TOC
Math
Eng
DLC
CN
```

4. critical section is a code segment where the shared variables can be accessed. An atomic action is required in a critical section. only one process can execute in its critical section at a time. All the other processes have to wait to execute in their critical section.

```
do {
    Entry section
    critical section
    Exit section
    Remainder section
} while (True)
```

In this diagram, the entry section handles the entry into the critical section. It acquires the resources needed for execution by the process. The exit section handles the exit from the critical section. It releases the resources and also informs the other processes that the critical section is free.

5. Race condition:

A race condition is a condition when there are many processes and every process shares the data with each other and accessing the data concurrently, and the output of execution depends on a particular sequence in which they share the data and access.

Example of race condition:

```
Void bankAccount (double money)
{
    shared = shared + money
}
```

Here, uses two variables. suppose shared is a shared variable.

Let's say that bankAccount function is called for its execution. The statements of this function will be executed in following sequence.

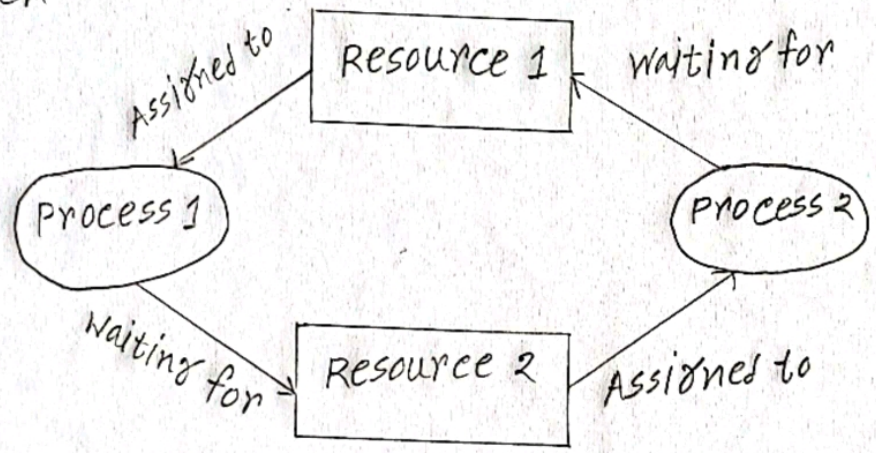
- The previous value of the shared variable will be loaded into one of the registers of the CPU.
- The value of money variable will be loaded into some another register.
- The value of two variables will be stored in two registers and the result will be calculated.
- Now, assign the result to the variable share.

6. Security breach:

A security breach occurs when a network or system is accessed by an unauthorized individual or application. Once your system is infiltrated, the intruders can steal data, install viruses, and compromise software. Needless to say, a breach can be a complete disaster for a managed services provider and their customers. cybercrime seems to be growing more sophisticated with each passing day, and hackers are constantly adopting new techniques as they attempt to breach security measures. And there are some security breach those are,

- MAN-IN-THE-MIDDLE ATTACK
- Denial of service and Distributed Denial of service Attacks.
- Phishing and Spear Phishing
- Password Attacks.
- Eavesdrop Attacks.
- cross-site scripting Attacks.
- Malware Attacks.

7. Deadlock: Deadlock occurs when each process holds a resource and wait for other resource held by any other process. Necessary conditions for deadlock to occur are Mutual Exclusion. Hold and wait, No Preemption and circular wait. For Example, in the below diagram, process 1 is holding resource 1 and waiting for resource 2 which is acquired by process 2, and process 2 is waiting for resource 1. Hence both process 1 and process 2 are in deadlock.



Starvation: Starvation is the problem that occurs when high priority processes keep executing and low priority processes get blocked for indefinite time. In heavily loaded computer system, a steady stream of higher-priority processes can prevent a low-priority process from ever getting the CPU.

*Difference between Deadlock and Starvation:

Deadlock	Starvation
1. All processes keep waiting for each other to complete and none get executed.	1. High priority processes keep excuting and low priority processes are blocked.
2. Resource are blocked by the processes	2. Resources are continuously utilized by high priority processes
3. Necessary conditions Mutual Exclusion, Hold and wait, No preemption, circular wait	3. Priorities are assigned to the processes
4. It can be prevented by avoiding the necessary conditions for deadlock	4. It can be prevented by Aging.

8. Deadlock Avoid: The most prevention algorithms have poor resource utilization, and hence result in reduced throughputs. We can try to avoid deadlocks by making use prior knowledge about the usage of resources by processes including resources available, resources allocated, future requests and future releases by processes. Most deadlock avoidance algorithms need every process to tell in advance the maximum number of resources of each type that it may need. Based on all these info we may decide if a process should wait a resource or not, and thus avoid chances for circular wait.

Deadlock Prevent: Deadlock prevention algorithms ensure that at least one of the necessary conditions. Those are, Mutual exclusion, hold and wait, no Preemption and circular does not hold true. However most prevention algorithms have poor resource utilization, and hence result in reduced throughputs.

9.a] soln for the following problem with Banker algorithm;

for, A B C

P_3 process allocated 2 1 1

P_3 process need 0 1 ;

The available resource 2 3 0 [allocability]

∴ P_3 current allocation 2 1 1

so, P_3 done its exceution and release all resources and available resource 4 4 1

for,

P_2 process allocated 3 0 2

P_2 ,, need 0 0

P_2 current allocation 6 0 2 [available = 1 4 1]

so, P_2 done it's exceution and release all resources. and available resource 7 4 3

for, P_1 process allocated 3 0 2

P_1 ,, need 0 2 0

P_1 current allocation 3 2 2 [available = 7 2 3]

P.t.o

So, P_1 done Its execution and release all resources and Available resources 10 4 5

for,

P_4 Process allocated 0 0 2

P_4 " need 4 3 1

P_4 current allocation 4 3 2 [available = 6 1 5]

So, P_4 done Its execution and release resources and available resources 10 4 7

for,

P_0 Process allocated 0 1 0

P_0 " need 7 4 3

P_0 current allocation 7 4 3 [available = 3 1 4]

So, P_0 done Its execution and release all resources and Available = 10 5 7

∴ safe state is

A	B	C
10	5	7

2.B|

If P_0 request for (0,2,15) as our available resource is (2,3,0), so It win not granted immediately because it is not in range

— 0 —