Name : Nidhi Patel

201912046

Q1 Signal :

a. Every process will be terminated, because in kill () function call, pid is negative means, it will sent to every child of that parent's process group. So, they all have pgid = 10. So, all will be kill.

b.

```
if ( (p2 = fork()) == 0 )    // 2nd child
{
        printf ( " Child 2 " );
        setpgid ( getpid (), 12 );
}
```

→ If same kill will execute as (a.) then parent & child 1 will be kill & not child 2 because child 2 's pgid is changed.

c.

```
#include <signal.h>
#include <stdio.h>
#include <sys/wait.h>
#include <errno.h>
#include <sys/types.h>
#include <unistd.h>
```

SLTIET
आत्मदीपोभव

```
void alarmHandler ();
void (* oldhandler) ();
void disableCritical();

void disableCritical()
{
    printf (" Normal section");
    kill (getpgrp()*(-1), SIGUSR2);
    signal (SIGALRM, alarmHandler);
    alarm(5);
}

void alarmHander() {
    printf ("Critical section");
    kill (getpgrp ()* (-1), SIGUSR1);
    signal (SIGALRM, disableCritical);
    alarm(3);
}

void siguserhandler (int signum)
{
    switch (signum )
    {
        case SIGUSR1 :
            oldhandler = signal (SIGINT, SIG_IGN);
            break;
        case SIGUSR2 :
            signal (SIGINT, oldhandler);
            break;
    }
}
```

```
int main(void)
{
      int status;
      int pid = fork();
      if (pid! =0){
         signal (SIGUSR1, siguserhandler);
         wait (& status);
      }
      else
      {
           signal (SIGUSR1, siguserhandler);
           signal (SIGUSR2, siguserhandler);
           printf ("child ");
           signal (SIGALRM, alarmHandler);
           alarm (5);
           while (1);
      }
}
```

Name : Nidhi Patel

201912046

SLTIET
आत्मदीपोभव

Q2 Concurrent Programming :

a. Thread 0
Thread 1

Or Thread 1
Thread 0

→ Because, it is concurrent so sequence may differe.

b. Thread 2
Thread 2

Or Thread 1
Thread 2

→ Because, it is concurrent so output may differ

c. Thread 2
Thread 2

d. Thread 0
Thread 1

e.

Line 1 : sem_init (&mutex ,0, 1);
Line 2 : sem_int (&items, 0, 1);
Line 3 : sem_int (&slots, 0, 1);

Line 4 : sem_wait (&items);
Line 5 : sem_wait (&mutex);

Line 6 : sem_post (&items);
Line 7 : sem_post (&mutex);

Line 8 : sem_wait (&items);
Line 9 : sem_wait (&slots);

Line 10 : sem_post (&items);

Line 11 : sem_post (&slots);

Name : Nidhi Patel
201912046

Q3 Network :

a. Client	Server
Socket	socket bind listen
Connect	accept

b. Socket : It will use to create socket file descriptor It will create on both client & server side.
- It is end point of Connection
- And Socketfd is listenfd for Server Side & Clientfd for Client side

~~Bind : It will bind the socket address of server to file descriptor that~~

Bind : It will bind the listenfd (Server side) to the server socket address.

listenfd : Listenfd is where the server is waiting for the client request & ~~so~~ as soon as the request is accepte accept () call will create a new fd

Name : Nidhi Patel

201912046

accept : As soon as request is accepted from client it will create a new file descriptor called connfd. Blocking call

connect : It will send / connection request to

connect : It will communicate between client & server with all details.

c.

```
main(){

    int cb = 0 ; int cur = 0;
    int port = 500;

    while (cb<4) {
          cb ++ ;
          getaddrinfo (NULL, port, & hint,
                                    & listf );
          listenfd = socket ( listP -> ai_family,
                              listp -> ai_socktype,
                              listp -> ai_protocole);
          bind( listenfd, 500);
          while(1) {
              connfd = accept ( listenfd,
                       (struct socket_addr), & clientaddr,
                       & clientlen);
              cur++ ;
```

```
        pthread_create( &tid , NULL, Handle_Client,
                                          connfd );
        if ( cur == 500 ) {
             ~~cur~~
             cur = 0 ;
             ~~port~~ cb ++ ;
             break ; }
      }
  }
}
```

Name : Nidhi Patel

201912046

Page ; 9 of 9

SLTIET
आत्मदीपोभव

Q4 Device Driver :

a. Major Number represents which driver will used to manage a particular device. It allows multiple devices to have the same major number if they are managed by the same driver.

Ex : there are 4 Hdevices represented as minor numbers 0, 10, 11, 12 by they are manage by same driver number 4.

```
9 ls -ltr /dev
crw--w---1 root +1 4, 0 +1 0
crw--w---1 root +1 4 , 10 +1 10
crw--w ---1 root +1 4 , 12 +1 12
crw--w ---1 root +1 4 , 11 +1 11
```

b. kmalloc is used to dynamically allocate memory to buffers in kernel code. Yes, we use this same function for user application by using ~~GFP_USR~~ GFP_USER.