1. Type and run the following program and see what it does.

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <signal.h>

int count = 0;

static void sigalarm_handler(int signo)
{
    printf("I am in interrupt handler (i.e. signal handler) \n");
    printf("bye bye. I serviced the interrupt \n\n");
    count++;
    return;
}

int main()
{
    int i;

    /* associate above function as the signal handler */
    signal(SIGALRM, sigalarm_handler);

    /* now set the timer using the alarm() system call */
    alarm(3);

    /* now loop around */
    while (1)
    {
        printf("I am in while loop \n");
        if (count >= 1)
            break;
        usleep(100000);
    }
}
```
for (i=0; i<10; ++i) {
    printf("I am looping a few more times\n");
    usleep(100000);
}

exit(0);
}

Is there a race condition in the program above? How can a race condition occur? Can you modify the program in some way so that no race condition will occur?

2. Write a routine called sleep_routine() that will take one argument which is the time to sleep in seconds. When a process calls this routine, the process will sleep that specified amount of time. Use the alarm() system call and SIGALRM signal of Linux to implement this routine. Test the routine with a small program. Provide the source code of the program and routine (printed to paper) as part of the homework.

3. Assume we have a machine that has 4 page frames. Assume the virtual address space consists of 8 pages. We are given the following page reference string. Analyze the performance of the following page replacement algorithms with this string: a) LRU, b) FCFS, c) Least Frequently Used.

3 6 3 5 7 4 5 6 0 3 4 6 5 1 4 6 2 3 2 6 4 7

4. int a[1024][1024];
   void main()
   {
        int i, j;

        for (i = 0; i < 1024; ++i)
            for (j = 0; j < 1024; ++j)
                a[i][j] = i * j;
   }

Assume we have a system that has virtual memory system where page size is 1024 bytes. The above program is to be run in this system. Assume the text segment of the program is put into a single page. The data segment of the program which includes the global variable a is put into some number of consecutive pages of the virtual memory that is used by the program. Assume, when placing the elements of the array, the compiler uses an order where first the elements of row 1 are placed, then the elements of row 2 are placed, etc. In other words, first element a[0][0] is placed, then a[0][1], then a[0][2], ..., a[1][0], ..., and finally a[1023][1023].

Assume none of the pages of the program is present in main memory initially. They are loaded on demand. How many page faults will occur due to accesses to the data segment (i.e. to the array elements)? Assume an integer occupies 4 bytes of storage. Assume the program can use at most 256 page frames of physical memory and LRU page replacement algorithm is applied.

What about the following program? How many page faults will occur in this case?
int a[1024][1024];
void main()
{
    int i, j;

    for (i = 0; i < 1024; ++i)
        for (j = 0; j < 1024; ++j)
            a[j][i] = i * j;
}

5. Assume the following requests for cylinder accesses are queued at the disk driver of a computer system. Assume the head is initially at cylinder (track) 70 and the direction of motion is initially towards higher cylinder numbers.

95 30 110 8 67 125 129 80 20 140

Find out the order of servicing these requests when the following disk scheduling algorithms are applied: a) SSF, b) FCFS, c) Elevator Algorithm.

Find out also the total number of cylinder skips (the amount of motion) required to serve all these requests for each of the above-mentioned algorithms. As a convention, assume that the number of cylinder skips required for moving from a cylinder $X$ to a cylinder $Y$ is equal to $|Y - X|$. 

6. A mouse can be connected to a computer through various connection points: the serial port, the USB port, or the PS/2 port. Assume we connect a serial mouse to the computer via one of the serial ports of the computer. Then the following program in Linux can be used to read the input coming from the mouse and take the necessary actions to draw something on the screen. If you have a serial mouse, connect it to your computer and run the following program, and see what kind of input is received from the mouse. If you don't have a serial mouse skip this question. This question is not mandatory to answer. It is just for teaching you how we can read data from a serial port. The serial port does not have to be connected to mouse. It could be connected to any other device that is using serial communication like a digital camera, a robot, an RS232 terminal, etc.

```c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <stdio.h>
#include <strings.h>
#include <unistd.h>
#include <stdlib.h>

#define BAUDRATE B19200

/* #define BAUDRATE B38400 */
#define BAUDRATE B19200
```
#define SERIALPORT "/dev/ttyS0"
#define FALSE 0
#define TRUE 1

volatile int STOP = FALSE;

int main()
{
    int fd, res;
    struct termios oldtio, newtio;
    unsigned char buf[255];

    fd = open(SERIALPORT, O_RDWR | O_NOCTTY);
    if (fd <0)
        {perror(SERIALPORT); exit(-1); }

    tcgetattr(fd,&oldtio); /* save current port settings */

    bzero(&newtio, sizeof(newtio));
    newtio.c_cflag = BAUDRATE | CRTSCTS | CS8 | CLOCAL | CREAD;
    newtio.c_iflag = IGNPAR;
    newtio.c_oflag = 0;
    /* set input mode (non-canonical, no echo,...) */
    newtio.c_lflag = 0;
    newtio.c_cc[VTIME]   = 0; /* inter-character timer unused */
    newtio.c_cc[VMIN]   = 1; /* blocking read until 5 chars received */

    tcflush(fd, TCIFLUSH);
    tcsetattr(fd,TCSANOW,&newtio);
    while (1) {
        res = read(fd, buf, 1);
        if (res == 1) {
            printf("%X ", (unsigned char) buf[0]); fflush(stdout);
            if (buf[0]=='z')
                break;
        }
    }
    tcsetattr(fd,TCSANOW,&oldtio);

    exit(0);
}
7. Assume we have a hard disk which has the RPM parameter as 5200 revolutions per minute. Assume a platter on the hard disk has 10000 tracks. Assume the radius of a platter is 5 cm. Compute the relative linear speed of the head with respect to the track that is currently under the head:

   a) For track that has the radius = 5 cm.
   b) For track that has the radius = 2 cm.

If the amount of information (number of bytes or sectors) stored on a track is directly related with the physical length of the track, and if the track at radius 2 cm stores 100 sectors, how many sectors can be stored at the track that has the radius as 5 cm? Assume all sectors has the size in terms of the number of bytes that can be stored in a sector.

8. Assume we have a 2.5 GHz machine. We would like the clock (timer) hardware to generate interrupts to the CPU every 20 ms (i.e. one clock tick is 20 ms). For this to happen, what value should be stored in the holding register of the clock hardware (this value will then be copied to the counter and the counter will count down to zero) by the driver of the clock?

9. A 64-bit computer system is using 3 level page tables. Assume, each page table entry occupies 8 bytes. The first level page table is indexed by 16 bits, the second level page table is indexed by 16 bits and the third level page is indexed by 20 bits. Assume we have 512 MB of RAM.

Answer the following questions based on the given information:

   a) What is the page size?
   b) How many page frames can be there in the RAM at most?
   c) How many top level page tables do we need for a program that is 256 MB?
   d) How many third level page tables do we need for a program that is 256 MB?
   e) What is the space (in bytes) required in RAM to store the top level page table? (We store a page table as a whole (not partially)).

10. Assume we have the following page table for a machine where a process may have 16 pages at most. Assume the RAM in the machine has 8 page frames. Assume the page size is 4 KB.
For each of the following virtual address, compute the respective physical addresses where data can be retrieved from if the page is in memory. If the page that is holding the specified virtual address is not in memory, denote this as a page fault.

Virtual Addresses: 9756 493125 57890 27950 37690 16900 21312

<table>
<thead>
<tr>
<th>virtual page number</th>
<th>page frame number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>x</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>x</td>
</tr>
<tr>
<td>9</td>
<td>x</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>x</td>
</tr>
<tr>
<td>12</td>
<td>x</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>x</td>
</tr>
</tbody>
</table>

11. Assume we have a single big-platter disk that has 10 tracks. Assume track 0 is located at 5 cm from the center, and track 9 is located at 2 cm from the center. Assume the number of sectors that a track has is directly proportional with the physical length of the track (i.e. if we consider a track as a circle with radius \( R \), than the information stored in the track is related with \( 2 \pi R \)). Assume the innermost track has 10 sectors.

Assume the operating system request for sectocs (blocks) in a uniform manner. This means, if there are a total of \( N \) sectors on the disk, the probability that a sector \( i (0 \leq i \leq N - 1) \) will be requested is the same for all the sectors.

Assume the seek time required to go from a track \( X \) to a track \( Y \) is directly proportional with \( |X - Y| \). Assume also that the seek time is 1 ms if \( |X - Y| = 1 \).

Answer the following questions based on this information:

a) What is the maximum seek time.

b) What is the average seek time (or more precisely, expected seek time) considering the uniform access pattern of the operating system to the sectors of the disk.

12. Assume a hard disk has 5000 cylinders (numbered from 0 to 4999), 8 tracks per cylinder (numbered from 0 to 7), and 200 sectors per track (numbered from 0 to 199). For all tracks of the disk, the number of sectors per track is the same.

Sequential data is placed on the disk starting from cylinder 0. When cylinder 0 is filled up, cylinder 1 is used, and so on.
Assume each sector of the hard disk is structured the following way. The sector starts with a 6 bytes preamble field, then comes 512 bytes data field, and finally the sector ends with 16 bytes ECC field. Assume the intersector gap is 4 bytes between any two sectors on a track (remember a track has a circular shape).

Answer the following questions.

a) What is the total number of sectors in the disk?

b) What is the raw maximum capacity of the disk? During this calculation, you have to include the preamble and ECC fields of a sector in the capacity.

c) What is the effective maximum capacity of the disk? During this calculation, you should not include the preamble and ECC fields of a sector in the capacity.

d) Find out the (cylinder, track, sector) parameter values for the following logical block numbers (assume each block is 512 bytes) that can be referred by operating system and that have to be translated to the actual cylinder, track and sector numbers on the disk. Assume block number 0 corresponds to cylinder 0, track 0, sector 0.

Logical block numbers: 356902, 20000, 1250896, 1004, 3567721.