```
1) #include <unistd.h>
int main() {
    int nread;
    char buff[20];
    nread = read(0, buff, 10);   // read 10 bytes from keyboard
    write(1, buff, nread);       // write the read bytes to screen
}

Commands:
cc  filename.c
./a.out
Pesce mandya
Pesce mandya


2) #include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<stdio.h>
int main() {
    int n,f,f1;
    char buff[10];
    f=open("seeking",O_RDWR);
    f1=lseek(f,10,SEEK_SET);
    printf("Pointer is at %d position\n",f1);
    read(f,buff,10);
    write(1,buff,10);
}

3.0)Shared Memory for Writer Process
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
#include<string.h>
int main() {
    int i;
    void *shared_memory;
    char buff[100];
    int shmid;
    shmid=shmget((key_t)2345, 1024, 0666|IPC_CREAT);
    printf("Key of shared memory is %d\n",shmid);
    shared_memory=shmat(shmid,NULL,0);
    printf("Process attached at %p\n",shared_memory);
    printf("Enter some data to write to shared memory\n");
    read(0,buff,100);
    strcpy(shared_memory,buff);
    printf("You wrote : %s\n",(char *)shared_memory);
}

3.1)Shared Memory for Reader Process
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
#include<string.h>
int main() {
    int i;
    void *shared_memory;
    char buff[100];
    int shmid;
    shmid=shmget((key_t)2345, 1024, 0666);
    printf("Key of shared memory is %d\n",shmid);
    shared_memory=shmat(shmid,NULL,0);
    printf("Process attached at %p\n",shared_memory);
    printf("Data read from shared memory is : %s\n",(char *)shared_memory);
}
```

4)FIFO

```
#include<stdio.h>
#include<conio.h>
int main() {
    int i, j, k, f, pf=0, count=0, rs[25], m[10], n;
    clrscr();
    printf("\n Enter the length of reference string -- ");
    scanf("%d",&n); printf("\n Enter the reference string -- ");
    for(i=0;i<n;i++)
    scanf("%d",&rs[i]);
    printf("\n Enter no. of frames -- ");
    scanf("%d",&f);
    for(i=0;i<f;i++)
    m[i]=-1;
    printf("\n The Page Replacement Process is -- \n");
    for(i=0;i<n;i++) {
        for(k=0;k<f;k++) {
            if(m[k]==rs[i]) break;
        }
        if(k==f) {
            m[count++]=rs[i];
            pf++;
        }
        for(j=0;j<f;j++)
        printf("\t%d",m[j]);
        if(k==f)
        printf("\tPF No. %d",pf); printf("\n");
        if(count==f)
        count=0;
    }
    printf("\n The number of Page Faults using FIFO are %d",pf);
    getch();
}
```

5)LRU

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int i, j, k, min, rs[25], m[10], count[10], flag[25], n, f, pf = 0, next =
1;
    clrscr();
    printf("Enter the length of reference string -- ");
    scanf("%d", &n);
    printf("Enter the reference string -- ");
    for (i = 0; i < n; i++) {
        scanf("%d", &rs[i]);
        flag[i] = 0;
    }
    printf("Enter the number of frames -- ");
    scanf("%d", &f);
    for (i = 0; i < f; i++) {
        count[i] = 0;
        m[i] = -1;
    }
    printf("\nThe Page Replacement process is -- \n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < f; j++) {
            if (m[j] == rs[i]) {
                flag[i] = 1;
                count[j] = next;
                next++;
            }
        }
        if (flag[i] == 0) {
            if (i < f) {
                m[i] = rs[i];
                count[i] = next;
                next++;
            } else {
                min = 0;
                for (j = 1; j < f; j++) {
                    if (count[min] > count[j])
                        min = j;
                }
                m[min] = rs[i];
                count[min] = next;
                next++;
            }
            pf++;
        }
        for (j = 0; j < f; j++)
            printf("%d\t", m[j]);
        if (flag[i] == 0)
            printf("PF No. -- %d", pf);
        printf("\n");
    }
    printf("\nThe number of page faults using LRU are %d", pf);
    getch();
}
```

6)SSTF

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
int main() {
    int queue[100], t[100], head, seek = 0, n, i, j, temp;
    float avg;
    clrscr();
    printf("*** SSTF Disk Scheduling Algorithm ***\n");
    printf("Enter the size of Queue\t");
    scanf("%d", &n);
    printf("Enter the Queue\t");
    for (i = 0; i < n; i++) {
        scanf("%d", &queue[i]);
    }
    printf("Enter the initial head position\t");
    scanf("%d", &head);
    for (i = 1; i < n; i++)
        t[i] = abs(head - queue[i]);
    for (i = 0; i < n; i++) {
        for (j = i + 1; j < n; j++) {
            if (t[i] > t[j]) {
                temp = t[i];
                t[i] = t[j];
                t[j] = temp;
                temp = queue[i];
                queue[i] = queue[j];
                queue[j] = temp;
            }
        }
    }
    for (i = 1; i < n - 1; i++) {
        seek = seek + abs(head - queue[i]);
        head = queue[i];
    }
    printf("\nTotal Seek Time is%d\t", seek);
    avg = seek / (float)n;
    printf("\nAverage Seek Time is %f\t", avg);
    return 0;
}
```

7)SCAN

```
#include <stdio.h>
int main() {
    int t[20], d[20], h, i, j, n, temp, atr[20], sum = 0, p = 0;
    printf("Enter the no of tracks to be traveresed: ");
    scanf("%d", &n);
    printf("Enter the position of head: ");
    scanf("%d", &h);
    t[0] = 0;
    t[1] = h;
    printf("Enter the tracks: ");
    for (i = 2; i < n + 2; i++)
        scanf("%d", &t[i]);
    for (i = 0; i < n + 2; i++) {
        for (j = 0; j < n + 1; j++) {
            if (t[j] > t[j + 1]) {
                temp = t[j];
                t[j] = t[j + 1];
                t[j + 1] = temp;
            }
        }
    }
    for (i = 0; i < n + 2; i++)
        if (t[i] == h)
            j = i;
    for (i = j; i >= 0; i--)
        atr[p++] = t[i];
    for (i = j + 1; i < n + 2; i++)
        atr[p++] = t[i];
    for (i = 0; i < p - 1; i++) {
        d[i] = (atr[i] > atr[i + 1]) ?
                atr[i] - atr[i + 1] :
                atr[i + 1] - atr[i];
        sum += d[i];
    }
    printf("\nAverage header movements: %f", (float)sum / n);
    return 0;
}
```

8)Sequential

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
struct fileTable {
    char name[20];
    int sb, nob;
} ft[30];
void main() {
    int i, j, n;
    char s[20];
    clrscr();
    printf("Enter no of files: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        printf("\nEnter file name %d: ", i + 1);
        scanf("%s", ft[i].name);
        printf("Enter starting block of file %d: ", i + 1);
        scanf("%d", &ft[i].sb);
        printf("Enter no of blocks in file %d: ", i + 1);
        scanf("%d", &ft[i].nob);
    }
    printf("\nEnter the file name to be searched -- ");
    scanf("%s", s);
    for (i = 0; i < n; i++)
        if (strcmp(s, ft[i].name) == 0)
            break;
    if (i == n)
        printf("\nFile Not Found");
    else {
        printf("\nFILE NAME START BLOCK NO OF BLOCKS BLOCKS OCCUPIED\n");
        printf("\n%s\t\t%d\t\t%d\t", ft[i].name, ft[i].sb, ft[i].nob);
        for (j = 0; j < ft[i].nob; j++)
            printf("%d, ", ft[i].sb + j);
    }
    getch();
}
```

9)Indexed

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
struct fileTable {
    char name[20];
    int nob, blocks[30];
} ft[30];
void main() {
    int i, j, n;
    char s[20];
    clrscr();
    printf("Enter no of files: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        printf("\nEnter file name %d: ", i + 1);
        scanf("%s", ft[i].name);
        printf("Enter no of blocks in file %d: ", i + 1);
        scanf("%d", &ft[i].nob);
        printf("Enter the blocks of the file: ");
        for (j = 0; j < ft[i].nob; j++)
            scanf("%d", &ft[i].blocks[j]);
    }
    printf("\nEnter the file name to be searched -- ");
    scanf("%s", s);
    for (i = 0; i < n; i++)
        if (strcmp(s, ft[i].name) == 0)
            break;
    if (i == n)
        printf("\nFile Not Found");
    else {
        printf("\nFILE NAME NO OF BLOCKS BLOCKS OCCUPIED");
        printf("\n %s\t\t%d\t", ft[i].name, ft[i].nob);
        for (j = 0; j < ft[i].nob; j++)
            printf("%d, ", ft[i].blocks[j]);
    }
    getch();
}
```