Q.11.Suppose that a disk drive has 5 ,000 cylinders, numbered 0to499 9.The drive is currently serving a request at cylinder 143 , and the previous request was at cylinder 125.The queue of pending requests, in FIFO Order, is:86 , 1470, 913 ,1774, 948,1509,1022, 1750, 130 Starting from the current head position, what is the total distance(in cylinders) that the disk arm moves to satisfy all the pending requests for each of the SCAN disk-scheduling algorithms?

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int headposition,i,n;
   printf("\n Enter head position");
   scanf("%d",&headposition);
   printf("\nEnter number of requests");
   scanf("%d",&n);
   int req[n];
   for(i=0;i<n;i++)
   {
         scanf("%d",&req[i]);
   }
   int difrf=req[0]-head;
   if(diffr<0)
   {
          diffr=diffr*-1;
   }
   for(i=1;i<n1;i++)
   {
```

```
        if((req[i]-req[i-1])>0)
          diffr=diffr+(req[i]-req[i-1]);
        else
          diffr=diffr+(req[i-1]-req[i]);
   }
   printf("Seek time = %d\n",diff);
   getch();
}
```

Q14.If a teacher is being served at thef ood mess and during he period when he is being served, another teacher comes, then that teacher would get the service (food) next. This process might continue leading to increase in waiting time of students to get food. Ensure in your program that the waiting time of students is minimized.

```
#include<stdio.h>
int main()
{
   int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
   printf("Enter Total Number of Process:");
   scanf("%d",&n);

   printf("\nEnter Burst Time and Priority\n");
   for(i=0;i<n;i++)
   {
      printf("\nP[%d]\n",i+1);
      printf("Burst Time:");
      scanf("%d",&bt[i]);
      printf("Priority:");
      scanf("%d",&pr[i]);
```

```
    p[i]=i+1;          //contains process number
}

//sorting burst time, priority and process number in ascending order using selection sort
for(i=0;i<n;i++)
{
    pos=i;
    for(j=i+1;j<n;j++)
    {
        if(pr[j]<pr[pos])
            pos=j;
    }

    temp=pr[i];
    pr[i]=pr[pos];
    pr[pos]=temp;

    temp=bt[i];
    bt[i]=bt[pos];
    bt[pos]=temp;

    temp=p[i];
    p[i]=p[pos];
    p[pos]=temp;
}
```

```
wt[0]=0;        //waiting time for first process is zero

//calculate waiting time
for(i=1;i<n;i++)
{
   wt[i]=0;
   for(j=0;j<i;j++)
      wt[i]+=bt[j];

   total+=wt[i];
}

avg_wt=total/n;     //average waiting time
total=0;

printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{
   tat[i]=bt[i]+wt[i];    //calculate turnaround time
   total+=tat[i];
   printf("\nP[%d]\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
}

avg_tat=total/n;    //average turnaround time
```

```
    printf("\n\nAverage Waiting Time=%d",avg_wt);
    printf("\nAverage Turnaround Time=%d\n",avg_tat);


        return 0;
}
```