Q write a program in c to implement

1) First come First serve scheduling

```
#include<stdio.h>
int main(){
 int n, bt[20], wt[20], tat[20], awt=0, atat=0, i, j;
 printf("Enter total number of processes:");
 scanf("%d", &n);
 printf("\n Enter Process Burst time");
 for(i=0; i<n; i++)
 {
  printf("P[%d]:", i+1);
  scanf("%d", &bt[i]);
 }
 wt[0]=0;
 for(i=1; i<n; i++)
 {
  wt[i]=0;
  for(j=0; j<i; j++)
   wt[i] += bt[j];
 }
 printf("\n Process\t Burst Time\t waiting Time\t Turn around
                                        Time");
 for(i=0; i<n; i++)
 {
  tat[i]=bt[i]+wt[i];
  awt += wt[i];
  atat += tat[i];
  printf("\n P[%d]\t %d\t %d\t %d", i+1, bt[i], wt[i],
                                        tat[i];
}
```

```
awut /= i;
awtat /= i;
printf("\n Average Waiting Time : %d", awut);
printf("\n Average Turnaround Time : %d", awtat);
return 0;
}
```

Output :- Enter total number of process : 3
Enter Process Burst Time
P[1] : 33
P[2] : 2
P[3] : 1

Process	Burst Time	waiting Time	Turnaround Time
P[1]	33	0	33
P[2]	2	33	35
P[3]	1	35	36

Average waiting Time : 22
Average Turnaround Time : 34

2) SBortest Job First Scheduling

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

```
printf("\n Enter Burst time : \n");
for(i=0; i<n; i++)
{
 printf("p%d:", i+1);
 scanf("%d", &bt[i]);
 p[i] = i+1;
}
 for(i=0; i<n; i++)
{
  pos = i;
  for(j=i+1; j<n; j++)
{
  if(bt[j] < bt[pos])
  pos = j;
}
  temp = bt[i];
 bt[i] = bt[pos];
 bt[pos] = temp;
 temp = p[i];
 p[i] = p[pos];
 p[pos] = temp;
}
  wt[0] = 0;
 for(i=1; i<n; i++)
{
 wt[i] = 0;
 for(j=0; j<i; j++)
{
 wt[i] += bt[j];
 total += wt[i];
 }
}
```

```
avg_wt = (float) total/n;
total = 0;
printf ("\n Process \t Burst Time \t Waiting Time \t Turnaround
                                                    Time");

for(i=0; i<n; i++)
{
tat[i] = bt[i] + wt[i];
total = tat[i];
printf ("\n p %d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i],
                                                    tat[i];

}
avg_tat = (float) total/n;
printf ("\n\n Average Waiting Time = %f", avg_wt);
printf ("\n Average Turnaround Time = %f \n", avg_tat);
return 0;
}
```

Output:- Enter number of process : 5

Enter Burst Time:

p_1 : 4

p_2 : 3

p_3 : 7

p_4 : 1

p_5 : 2

Process	Burst Time	Waiting Time	Turnaround T
P_4	1	0	1
P_5	2	1	3
P_2	3	3	6
P_1	4	6	10
P_3	7	10	17

Average waiting Time = 4.000000
Average Turnaround Time = 7.400000

3.) Priority Scheduling

```
#include<stdio.h>
int main(){
int x, n, p[10], pp[10], pt[10], wt[10], tat[10], awt, atat, i;
printf("Enter the number of process:");
scanf("%d", &n);
printf("\n Enter process:\n");
for(i=0; i<n; i++)
{
printf("\n process no. of %d?", i+1);
scanf("%d %d", &pt[i], &pp[i]);
p[i] = i+1;
}
for(i=0; i<n-1; i++)
{
for(int j=i+1; j<n; j++)
{
if(pp[i] < pp[j])
{
x = pp[i];
pp[i] = pp[j];
pp[j] = x;
x = pt[i];
pt[i] = pt[j];
pt[j] = x;
x = p[i];
p[i] = p[j];
```

```
        p[j] = x;
      }
    }
  }
  w[0] = 0;
  awt = 0;
  t[0] = pt[0];
  atat = t[0];
  for (i=1; i<n; i++)
  {
    w[i] = t[i-1];
    awt = w[i];
    t[i] = w[i] + pt[i];
    atat = t[i];
  }
  printf ("\n\n Job \t Burst Time \t waiting time \t Turn
                                        Time Priority \n");

  for (i=0; i<n; i++)
  {
    printf ("\n %d \t\t %d \t\t %d \t\t %d \n", p[
                                        pt[i], w[i], t[i], pp[

  awt /= n;
  atat /= n;
  printf ("\n Average waiting Time: %d \n", awt);
  printf ("\n Average Turn Around Time: %d \n" atat
  return 0;
  }
```

Output:- Enter the number of process: 4

Enter process:
Process no. 1: 3

1
Process no 2: 4
2
Process no. 3: 5
3
Process no 4: 6
4

Job	Burst Time	Waiting Time	Turnaround Time	Priority
4	6	0	6	4
3	5	6	11	3
2	4	11	15	2
1	3	15	18	1

Average waiting Time: 8
Average Turn Around Time: 12

4.) Round Robin Scheduling

```
#include <stdio.h>
int main() {
  int i, limit, total=0, x, counter=0;
  int time_quantum;
  int wait_timing=0, turnaround_time=0;
  int arrival_time[10], burst_time[10], temp[10];
  float average_waiting_time, average_turnaround_time;
  printf("Enter total no. of processes :");
  scanf("%d", &limit);
  x = limit;
  for(i=0; i<limit; i++)
{
```

```
printf("\n Enter Details of Process [%d]", i+1);
printf("Arrival Time: t");
scanf("%d", &arrival_time[i]);
printf("Burst Time: t");
scanf("%d", &burst_time[i]);
temp[i] = burst_time[i];
}
printf("\n Enter Time quantum: t");
scanf("%d", &time_quantum);
printf("\n Process ID tt Burst Time t Turn around Time t
                                    waiting Time \n");

for(total=0; i=0; x1=0)
{
 if (temp[i] <= time_quantum && temp[i] > 0)
{
 total = total + temp[i];
 temp p[i] = 0;
 counter = 1;
}
 else if (temp[i] > 0)
{
 temp[i] = temp[i] - time_quantum;
 total = total + time_quantum;
}
 if (temp[i] == 0 && counter = 1)
{
   x--;
printf("\n Process [%d] tt %d tt %d ttt %d", i+1, burst_time
                - arrival_time[i], total arrival_time -
                time[i]);
wait_time = wait_time + total - arrival_time - burst_time
```

```
turnar_time = turnar_time + total - arrival_time[i];
counter = 0;
}
if (i == limit - 1)
{
i = 0;
}
else if (arrival_time[i+1] <= total
{{{
else if i++;
}
else
{
i = 0;
}
}
average_wait_time = wait_time * 1.0 / limit;
average_turnar_time = turnar_time * 1.0 / limit;
printf("\n\n Average waiting Time : %f", average_wait_time);
printf("\n Average Turnaround Time : %f" average_turnar_time
return 0;
}
```

Output:- Enter Total no. of Process : 4
Enter details of Process [1]
Arrival Time : 0
Burst Time : 4

Enter Details of Process [2]
Arrival Time : 1
Burst Time : 7

Date :
Page No. :

Enter details of Process [3]
Arrival Time : 2
Burst Time : 5

Enter details of Process [4]
Arrival Time : 3
Burst Time : 6

Enter Time Quantum : 3

Process ID	Burst Time	Turnaround Time	Waiting Time
Process [1]	4	13	9
Process [3]	5	16	11
Process [4]	6	18	12
Process [2]	7	21	14

Average waiting Time : 11.500000
Average Turnaround Time : 17.000000