```
//PRiority
#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;
}

//RR
#include <iostream>
using namespace std;
struct Process {
 int id;
 int burst_time;
 int remaining_time;
};
int main() {
 int n, quantum;
 cout << "Enter the number of processes: ";
 cin >> n;
 cout << "Enter the time quantum (ms): ";
 cin >> quantum;
 Process processes[n];
 // Input burst times for each process
 for (int i = 0; i < n; i++) {
 processes[i].id = i + 1;
 cout << "Enter burst time for process " << processes[i].id << ": ";
 cin >> processes[i].burst_time;
 processes[i].remaining_time = processes[i].burst_time;
 }
 int current_time = 0;
 int turnaround_time[n] = {0};
 int waiting_time[n] = {0};
 int completed = 0;
 while (completed < n) {
 for (int i = 0; i < n; i++) {
 if (processes[i].remaining_time > 0) {
 if (processes[i].remaining_time <= quantum) {
 current_time += processes[i].remaining_time;
 turnaround_time[i] = current_time;
 processes[i].remaining_time = 0;
 completed++;
 } else {
 current_time += quantum;
 processes[i].remaining_time -= quantum;
 }
 }
 }
 }
waiting_time[0]=0;
 // Calculate waiting time
 for (int i = 0; i < n; i++) {
 waiting_time[i] = turnaround_time[i] - processes[i].burst_time;
 }
 // Display results
 cout << "Process\tBurst Time\tTurnaround Time\tWaiting Time\n";
 for (int i = 0; i < n; i++) {
 cout << processes[i].id << "\t" << processes[i].burst_time << "\t\t"
 << turnaround_time[i] << "\t\t" << waiting_time[i] << endl;
 }
 return 0;}
```

```c
//FCFS
#include <stdio.h>

int main() {
   int pid[15], bt[15], wt[15], n;
   printf("Enter the number of processes: ");
   scanf("%d", &n);
   printf("Enter process IDs of all the processes: ");
   for (int i = 0; i < n; i++) {
      scanf("%d", &pid[i]);
   }
   printf("Enter burst times of all the processes: ");
   for (int i = 0; i < n; i++) {
      scanf("%d", &bt[i]);
   }
   wt[0] = 0;
   for (int i = 1; i < n; i++) {
      wt[i] = bt[i - 1] + wt[i - 1];
   }
   printf("\nProcess ID\tBurst Time\tWaiting Time\tTurnaround Time\n");
   float total_wt = 0.0, total_tat = 0.0;
   for (int i = 0; i < n; i++) {
      int tat = bt[i] + wt[i];
      printf("%d\t\t%d\t\t%d\t\t%d\n", pid[i], bt[i], wt[i], tat);
      total_wt += wt[i];
      total_tat += tat;
   }
   printf("\nAverage Waiting Time = %.2f\n", total_wt / n);
   printf("Average Turnaround Time = %.2f\n", total_tat / n);
   return 0;
}
```

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

   printf("\nEnter 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;

   printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
   for (i = 0; i < n; i++) {
      tat[i] = bt[i] + wt[i];
      totalT += 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 = (float)totalT / n;
   printf("\n\nAverage Waiting Time=%f", avg_wt);
   printf("\nAverage Turnaround Time=%f", avg_tat);

   return 0;
}
```

```
//PageFault=FIFO

#include <stdio.h>
int main() {
   int incomingStream[] = {4, 1, 2, 4, 5};
   int pageFaults = 0;
   int frames = 3;
   int m, n, s, pages;
   pages = sizeof(incomingStream) / sizeof(incomingStream[0]);
   printf(" Incoming \t Frame 1 \t Frame 2 \t Frame 3 ");
   int temp[frames];
   for (m = 0; m < frames; m++) {
      temp[m] = -1;
   }
   for (m = 0; m < pages; m++) {
      s = 0;
      for (n = 0; n < frames; n++) {
         if (incomingStream[m] == temp[n]) {
            s++;
            pageFaults--;
         }
      }
      pageFaults++;
      if ((pageFaults <= frames) && (s == 0)) {
         temp[m] = incomingStream[m];
      } else if (s == 0) {
         temp[(pageFaults - 1) % frames] = incomingStream[m];
      }
      printf("\n");
      printf("%d\t\t\t", incomingStream[m]);
      for (n = 0; n < frames; n++) {
         if (temp[n] != -1)
            printf(" %d\t\t\t", temp[n]);
         else
            printf(" - \t\t\t");
      }
   }
   printf("\nTotal Page Faults:\t%d\n", pageFaults);
   return 0;
}
```