**3.Design a CPU scheduling program with C using First Come First Served technique with the following considerations.**

**a. All processes are activated at time 0.**

**b. Assume that no process waits on I/O devices.**

1. #include <stdio.h>
2. int main()
3. {
4. int pid[15];
5. int bt[15];
6. int n;
7. printf("Enter the number of processes: ");
8. scanf("%d",&n);
10. printf("Enter process id of all the processes: ");
11. for(int i=0;i<n;i++)
12. {
13. scanf("%d",&pid[i]);
14. }
16. printf("Enter burst time of all the processes: ");
17. for(int i=0;i<n;i++)
18. {
19. scanf("%d",&bt[i]);
20. }
22. int i, wt[n];
23. wt[0]=0;
25. *//for calculating waiting time of each process*
26. for(i=1; i<n; i++)
27. {
28. wt[i]= bt[i-1]+ wt[i-1];
29. }
31. printf("Process ID Burst Time Waiting Time TurnAround Time**\n**");
32. float twt=0.0;
33. float tat= 0.0;
34. for(i=0; i<n; i++)
35. {
36. printf("%d**\t\t**", pid[i]);
37. printf("%d**\t\t**", bt[i]);
38. printf("%d**\t\t**", wt[i]);
40. *//calculating and printing turnaround time of each process*
41. printf("%d**\t\t**", bt[i]+wt[i]);
42. printf("**\n**");
44. *//for calculating total waiting time*
45. twt += wt[i];
47. *//for calculating total turnaround time*
48. tat += (wt[i]+bt[i]);
49. }
50. float att,awt;
52. *//for calculating average waiting time*
53. awt = twt/n;
55. *//for calculating average turnaround time*
56. att = tat/n;
57. printf("Avg. waiting time= %f**\n**",awt);
58. printf("Avg. turnaround time= %f",att);
59. }

**Program Explanation**

1. Initialize two array **pid[]** and **bt[]** of size 15.  
2. Ask the user for number of processes **n**.  
3. Ask the user for **process id** and **burst time** for all n processes and store them into **pid[]** and **bt[]** respectively.  
4. Calculate waiting time of each process by the formula **wt[i] = wt[i-1] + bt[i-1]**.  
5. Print Process Id, Burst Time, waiting time and Turnaround time of each process in tabular manner.  
6. Calculate and print turnaround time of each process = **bt[i] + wt[i]**.  
7. Add waiting time of all the processes and store it in the variable **twt**.  
8. Add turnaround time of all the processes and store it in the variable **tat**.  
9. Calculate average waiting time as **awt = twt/n**.  
10. Calculate average turnaround time as **att = tat/n**;  
11. Print **average waiting time** and **average turnaround time**.  
12. Exit.

**4.Construct a scheduling program with C that selects the waiting process with the smallest execution time to execute next**

#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;

}

//sorting of burst times

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;

//finding the waiting time of all the processes

for(i=1;i<n;i++)

{

wt[i]=0;

for(j=0;j<i;j++)

//individual WT by adding BT of all previous completed processes

wt[i]+=bt[j];

//total waiting time

total+=wt[i];

}

//average waiting time

avg\_wt=(float)total/n;

printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");

for(i=0;i<n;i++)

{

//turnaround time of individual processes

tat[i]=bt[i]+wt[i];

//total turnaround time

totalT+=tat[i];

printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

}

//average turnaround time

avg\_tat=(float)totalT/n;

printf("\n\nAverage Waiting Time=%f",avg\_wt);

printf("\nAverage Turnaround Time=%f",avg\_tat);

}

**PROGRAM EXPLANATION**

1. Initialize two array **pid[]** and **bt[]** of size 20.  
   2. Ask the user for number of processes **n**.  
   3. Ask the user for process id and burst time for all **n** processes and store them into **pid[]** and **bt[]** respectively.  
   4. Sort all the processes according to their burst time.  
   5. Assign waiting **time = 0** to the smallest process.  
   6. Calculate waiting time of each process by using two loops and adding all the burst time of previously completed processes.  
   7. Print Process Id, Burst Time, waiting time and Turnaround time of each process in tabular manner.  
   8. Calculate and print turnaround time of each process = **bt[i] + wt[i]**.  
   9. Add waiting time of all the processes and store it in the variable **total**.  
   10. Add turnaround time of all the processes and store it in the variable **totalT**.  
   11. Calculate average waiting time as **avg\_wt = total/n**.  
   12. Calculate average turnaround time as **avg\_tat = totalT/n**;  
   13. Print average waiting time and average turnaround time.  
   14. Exit.

**5.Construct a scheduling program with C that selects the waiting process with the highest priority to execute next.**

#include <stdio.h>

//Function to swap two variables

void swap(int \*a,int \*b)

{

int temp=\*a;

\*a=\*b;

\*b=temp;

}

int main()

{

int n;

printf("Enter Number of Processes: ");

scanf("%d",&n);

// b is array for burst time, p for priority and index for process id

int b[n],p[n],index[n];

for(int i=0;i<n;i++)

{

printf("Enter Burst Time and Priority Value for Process %d: ",i+1);

scanf("%d %d",&b[i],&p[i]);

index[i]=i+1;

}

for(int i=0;i<n;i++)

{

int a=p[i],m=i;

//Finding out highest priority element and placing it at its desired position

for(int j=i;j<n;j++)

{

if(p[j] > a)

{

a=p[j];

m=j;

}

}

//Swapping processes

swap(&p[i], &p[m]);

swap(&b[i], &b[m]);

swap(&index[i],&index[m]);

}

// T stores the starting time of process

int t=0;

//Printing scheduled process

printf("Order of process Execution is\n");

for(int i=0;i<n;i++)

{

printf("P%d is executed from %d to %d\n",index[i],t,t+b[i]);

t+=b[i];

}

printf("\n");

printf("Process Id Burst Time Wait Time TurnAround Time\n");

int wait\_time=0;

for(int i=0;i<n;i++)

{

printf("P%d %d %d %d\n",index[i],b[i],wait\_time,wait\_time + b[i]);

wait\_time += b[i];

}

return 0;

}

**PROGRAM EXPLANATION**

1. First, enter the total number of processes and store it in variable **n**.  
   2. After that, provide the burst time and priority and store it in variable **b** and **p**.  
   3. Finding out highest priority element and placing it at its desired position.  
   4. Sort the processes on the basis of the priority.  
   5. After that print the processed with their time stamp (starting time and ending time). Variable **T** stores the starting time of process.  
   6. In the end, print the **waiting time** and **turnaround time** for each process. Waiting time is the time spent in the ready queue, while turnaround time is the total time taken by process **(burst time + waiting time)**

**6. Construct a c program to implement pre-emptive priority scheduling algorithm.**

#include <stdio.h>

//Function to swap two variables

void swap(int \*a,int \*b)

{

int temp=\*a;

\*a=\*b;

\*b=temp;

}

int main()

{

int n;

printf("Enter Number of Processes: ");

scanf("%d",&n);

// b is array for burst time, p for priority and index for process id

int b[n],p[n],index[n];

for(int i=0;i<n;i++)

{

printf("Enter Burst Time and Priority Value for Process %d: ",i+1);

scanf("%d %d",&b[i],&p[i]);

index[i]=i+1;

}

for(int i=0;i<n;i++)

{

int a=p[i],m=i;

//Finding out highest priority element and placing it at its desired position

for(int j=i;j<n;j++)

{

if(p[j] > a)

{

a=p[j];

m=j;

}

}

//Swapping processes

swap(&p[i], &p[m]);

swap(&b[i], &b[m]);

swap(&index[i],&index[m]);

}

// T stores the starting time of process

int t=0;

//Printing scheduled process

printf("Order of process Execution is\n");

for(int i=0;i<n;i++)

{

printf("P%d is executed from %d to %d\n",index[i],t,t+b[i]);

t+=b[i];

}

printf("\n");

printf("Process Id Burst Time Wait Time TurnAround Time\n");

int wait\_time=0;

for(int i=0;i<n;i++)

{

printf("P%d %d %d %d\n",index[i],b[i],wait\_time,wait\_time + b[i]);

wait\_time += b[i];

}

return 0;

}

**program explanation**

 First, enter the total number of processes and store it in variable **n**.  
2. After that, provide the burst time and priority and store it in variable **b** and **p**.  
3. Finding out highest priority element and placing it at its desired position.  
4. Sort the processes on the basis of the priority.  
5. After that print the processed with their time stamp (starting time and ending time). Variable **T** stores the starting time of process.  
6. In the end, print the **waiting time** and **turnaround time** for each process. Waiting time is the time spent in the ready queue, while turnaround time is the total time taken by process **(burst time + waiting time)**.

**Output**

Enter Number of Processes: 3

Enter Burst Time and Priority Value for Process 1: 10 2

Enter Burst Time and Priority Value for Process 2: 5 0

Enter Burst Time and Priority Value for Process 3: 8 1

Order of process Execution is

P1 is executed from 0 to 10

P3 is executed from 10 to 18

P2 is executed from 18 to 23

Process Id Burst Time Wait Time TurnAround Time

P1 10 0 10

P3 8 10 18

P2 5 18 23

**7. Construct a C program to implement non-preemptive SJF algorithm.**

**#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;**

**}**

**//sorting of burst times**

**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;**

**//finding the waiting time of all the processes**

**for(i=1;i<n;i++)**

**{**

**wt[i]=0;**

**for(j=0;j<i;j++)**

**//individual WT by adding BT of all previous completed processes**

**wt[i]+=bt[j];**

**//total waiting time**

**total+=wt[i];**

**}**

**//average waiting time**

**avg\_wt=(float)total/n;**

**printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");**

**for(i=0;i<n;i++)**

**{**

**//turnaround time of individual processes**

**tat[i]=bt[i]+wt[i];**

**//total turnaround time**

**totalT+=tat[i];**

**printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);**

**}**

**//average turnaround time**

**avg\_tat=(float)totalT/n;**

**printf("\n\nAverage Waiting Time=%f",avg\_wt);**

**printf("\nAverage Turnaround Time=%f",avg\_tat);**

**}**

**Program explanation**

 Initialize two array **pid[]** and **bt[]** of size 20.  
2. Ask the user for number of processes **n**.  
3. Ask the user for process id and burst time for all **n** processes and store them into **pid[]** and **bt[]** respectively.  
4. Sort all the processes according to their burst time.  
5. Assign waiting **time = 0** to the smallest process.  
6. Calculate waiting time of each process by using two loops and adding all the burst time of previously completed processes.  
7. Print Process Id, Burst Time, waiting time and Turnaround time of each process in tabular manner.  
8. Calculate and print turnaround time of each process = **bt[i] + wt[i]**.  
9. Add waiting time of all the processes and store it in the variable **total**.  
10. Add turnaround time of all the processes and store it in the variable **totalT**.  
11. Calculate average waiting time as **avg\_wt = total/n**.  
12. Calculate average turnaround time as **avg\_tat = totalT/n**;  
13. Print average waiting time and average turnaround time.  
14. Exit.

**Output**

Enter number of process:4

Enter Burst Time:

p1:5

p2:4

p3:12

p4:7

Process Burst Time Waiting Time Turnaround Time

p2 4 0 4

p1 5 4 9

p4 7 9 16

p3 12 16 28

Average Waiting Time=7.250000

Average Turnaround Time=14.250000