

1. First come first serve (FCFS)

Code:

```
#include<iostream>
using namespace std;

int main()
{
    int n, bt[20], wt[20], tat[20], averageWaitingTime = 0,
    aveargeTurnAroundTime = 0;
    cout<<"Enter total number of processes(maximum 20):";
    cin>>n;

    cout<<"\nEnter Process Burst Time\n";
    for(int i=0;i<n;i++)
    {
        cout<<"P["<<i+1<<"]:";
        cin>>bt[i];
    }

    wt[0] = 0;
    for(int i= 1;i<n;i++)
    {
        wt[i] = 0;
        for(int j=0 ;j<i;j++)
            wt[i] += bt[j];
    }

    cout<<"\nProcess\t\tBurst Time\tWaiting
    Time\tTurnaround Time";

    //calculating turnaround time
    for(int i=0;i<n;i++)
```

```

{
    tat[i] = bt[i] + wt[i];
    averageWaitingTime += wt[i];
    aveargeTurnAroundTime += tat[i];

    cout<<"\nP["<<i+1<<"]"<<"\t\t"<<bt[i]<<"\t\t"<<wt[i]<<"\t\t"<
    <tat[i];
}

    averageWaitingTime /= n;
    aveargeTurnAroundTime /= n;
    cout<<"\n\nAverage Waiting Time:
"<<averageWaitingTime;
    cout<<"\nAverage Turnaround Time:
"<<aveargeTurnAroundTime;
    return 0;
}

```

Output:

The screenshot shows the Code::Blocks IDE with a C++ program running. The console window displays the following output:

```

Enter total number of processes(maximum 20):3
Enter Process Burst Time
P[1]:12
P[2]:34
P[3]:23

Process      Burst Time      Waiting Time      Turnaround Time
P[1]         12              0                12
P[2]         34              12               46
P[3]         23              46               69

Average Waiting Time: 19
Average Turnaround Time: 42
Process returned 0 (0x0)   execution time : 14.585 s
Press any key to continue.

```

The code editor shows the following lines of code:

```

37     aveargeTurnAroundTime += tat[i];
38     cout<<"\nP["<<i+1<<"]"<<"\t\t"<<bt[i]<<"\t\t"<<wt[i]<<"\t\t"<<tat[i];
39 }
40
41     averageWaitingTime /= n;
42     aveargeTurnAroundTime /= n;

```

2. Priority Scheduling

Code:

```

#include<bits/stdc++.h>
using namespace std;

```

```

struct process{
    int burstTime;
    int priority;
    int process_number;
};
typedef struct process *prc;
bool comp(prc A, prc B)
{
    return A->priority < B->priority;
}
int main()
{
    int n;
    cout<<"Enter Total Number of Process:";
    cin>>n;
    vector<prc>vec;
    cout<<"\nEnter Burst Time and Priority\n";
    for(int i=0;i<n;i++)
    {
        int a, b;
        cout<<"\nP["<<i+1<<"]\n";
        cout<<"Burst Time: ";
        cin>>a;
        cout<<"Priority:  ";
        cin>>b;
        prc p = new process;
        p->burstTime = a;
        p->priority = b;
        p->process_number = i + 1;
        vec.push_back(p);
        //contains process number
    }
    // less number, higher priority
    sort(vec.begin(), vec.end(), comp);

```

```

int wt[n], tat[n];
wt[0]=0;
int total = 0, pos, temp, avg_wt, avg_tat, i, j;
//waiting time for first process is zero

for(i=1;i<n;i++)
{
    wt[i]=0;
    for(j=0;j<i;j++)
        wt[i] += vec[j]->burstTime;

    total += wt[i];
}

avg_wt = total/n;
total = 0;

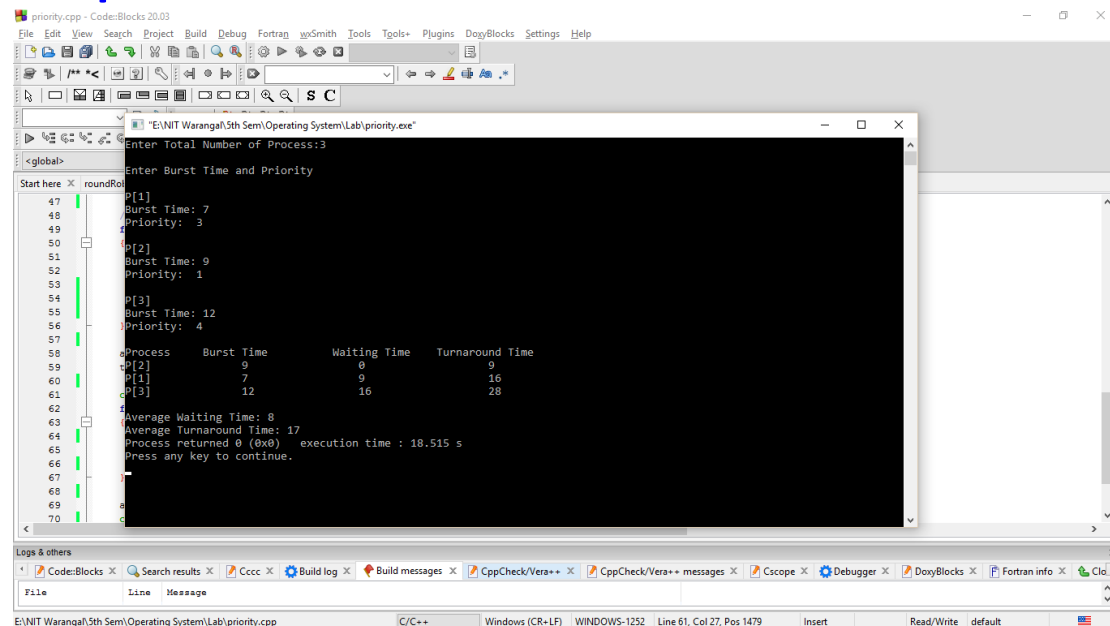
cout<<"\nProcess\t    Burst Time    \tWaiting
Time\tTurnaround Time";
for(i=0;i<n;i++)
{
    tat[i] = vec[i]->burstTime + wt[i];
    total += tat[i];
    cout<<"\nP["<<vec[i]->process_number<<"]\t\t
"<<vec[i]->burstTime<<"\t\t    "<<wt[i]<<"\t\t\t"<<tat[i];
}

avg_tat = total/n;    //average turnaround time
cout<<"\n\nAverage Waiting Time: "<<avg_wt;
cout<<"\nAverage Turnaround Time: "<<avg_tat;

return 0;
}

```

Output:



The screenshot shows a debugger window for a C++ program named 'priority.cpp'. The program is running, and the output is displayed in a console window. The output shows the execution of a priority scheduling algorithm for three processes (P[1], P[2], P[3]). The processes are entered with their burst times and priorities. The output then shows the execution order based on priority, with P[2] being the highest priority, followed by P[1] and then P[3]. The output also includes a table of process execution times, waiting times, and turnaround times, and finally calculates the average waiting time and average turnaround time.

```
Enter Total Number of Process:3
Enter Burst Time and Priority
P[1]
Burst Time: 7
Priority: 3
P[2]
Burst Time: 9
Priority: 1
P[3]
Burst Time: 12
Priority: 4
Process Burst Time Waiting Time Turnaround Time
P[2] 9 0 9
P[1] 7 9 16
P[3] 12 16 28
Average Waiting Time: 8
Average Turnaround Time: 17
Process returned 0 (0x0) execution time : 18.515 s
Press any key to continue.
```

3. Shortest Job First

Code:

```
#include<iostream>
using namespace std;
int main()
{
    int n, temp, tt = 0, m, d = 0, i, j;
    float atat = 0, awt = 0, stat = 0, swt = 0;

    cout<<"enter no of process:"<<endl;
    cin>>n;
    int a[n], b[n], e[n], tat[n], wt[n];

    for(i=0;i<n;i++)
    {
        cout<<"enter arrival time: ";
        cin>>a[i];
    }
    for(i=0;i<n;i++)
    {
```

```

        cout<<"enter burst time: ";
        cin>>b[i];
    }

    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++){
            if(b[i] > b[j])
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
                temp = b[i];
                b[i] = b[j];
                b[j] = temp;
            }
        }
    }
    m = a[0];
    for(i=0;i<n;i++)
    {
        if(m > a[i])
        {
            cout<<"hi";
            m = a[i];
            d = i;
        }
    }
    tt = m;
    e[d] = tt + b[d];
    tt = e[d];

    for(i=0;i<n;i++)
    {
        if(a[i] != m)

```

```

        {
            e[i] = b[i] + tt;
            tt = e[i];
        }
    }
    for(i=0;i<n;i++)
    {
        tat[i] = e[i] - a[i];
        stat = stat + tat[i];
        wt[i] = tat[i] - b[i];
        swt = swt + wt[i];
    }
    atat = stat/n;
    awt = swt/n;
    cout<<"Process   Arrival-time(s)   Burst-time(s)
Waiting-time(s)   Turnaround-time(s)\n";

    for(i=0;i<n;i++)
    {
        cout<<"P"<<i+1<<"
                                "<<a[i]<<"
        "<<b[i]<<"
                                "<<wt[i]<<"
        "<<tat[i]<<endl;
    }

    cout<<"average waiting time: "<<awt<<" average turn
around time: "<<atat;
    return 0;
}

```

Output:

```

1 // Enter no. of process:
2 // Enter arrival time: 1
3 // Enter arrival time: 2
4 // Enter burst time: 6
5 // Enter burst time: 7
6
7 <global> Process Arrival-time(s) Burst-time(s) Waiting-time(s) Turnaround-time(s)
8 Start here:
9 1 1 6 0 6
10 2 2 7 5 12
11
12 // Average waiting time: 2.5 average turn around time: 9
13 // Process returned 0 (0x0) execution time : 6.058 s
14 // Press any key to continue.
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51 for(i=0;i<n;i++)
52 {
53     if(a[i] != m)
54     {

```

4. Round Robin Algorithm

Code:

```
#include<iostream>
```

```
using namespace std;
```

```
// Function to find the waiting time for all
```

```
void findWaitingTime(int processes[], int n, int bt[], int wt[], int
quantum)
```

```
{
```

```
    int rem_bt[n];
```

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

```
        rem_bt[i] = bt[i];
```

```
    int t = 0; // Current time
```

```
    while (1) {
```

```
        bool done = true;
```

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

```
        {
```

```
            if (rem_bt[i] > 0)
```

```
            {
```

```
                done = false; // There is a pending process
```

```
                if (rem_bt[i] > quantum)
```



```

        {
            t += quantum;
            rem_bt[i] -= quantum;
        }
        else
        {
            t = t + rem_bt[i];
            wt[i] = t - bt[i];
            rem_bt[i] = 0;
        }
    }
}
if (done == true)
    break;
}
}

```

```

// Function to calculate turn around time
void findTurnAroundTime(int processes[], int n, int bt[], int wt[],
int tat[])
{
    for (int i = 0; i < n ; i++)
        tat[i] = bt[i] + wt[i];
}

```

```

// Function to calculate average time
void findavgTime(int processes[], int n, int bt[], int quantum)
{
    int wt[n], tat[n], total_wt = 0, total_tat = 0;
    findWaitingTime(processes, n, bt, wt, quantum);
    findTurnAroundTime(processes, n, bt, wt, tat);
    cout << "Processes "<< " Burst time "
        << " Waiting time " << " Turn around time\n";

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

```

```

    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        cout << " " << i+1 << "\t\t" << bt[i] << "\t " << wt[i] << "\t\t"
" << tat[i] << endl;
    }
    cout << "Average waiting time = " << (float)total_wt /
(float)n;
    cout << "\nAverage turn around time = " << (float)total_tat /
(float)n;
}
int main()
{
    int processes[] = { 4, 5, 6};
    int n = sizeof processes / sizeof processes[0];
    int burst_time[] = {10, 15, 20};
    int quantum = 3;
    findavgTime(processes, n, burst_time, quantum);
    return 0;
}

```

Output:

The screenshot shows the Code::Blocks IDE with the 'roundRobin.cpp' file open. The 'Debugger' window is active, displaying the output of the program. The output includes a table of process metrics and summary statistics.

Processes	Burst time	Waiting time	Turn around time
1	10	18	28
2	15	22	37
3	20	25	45

Average waiting time = 21.6667
Average turn around time = 36.6667
Process returned 0 (0x0) execution time : 0.018 s
Press any key to continue.