OPERATING SYSTEM PRACTICAL FILE

SUBMITTED BY :- ASHISH KUMAR

COURSE :-BSC.(H) COMPUTER SCIENCE

YEAR :- 2ND

SEMESTER :- 3RD

ROLL NO. :- 2K21/CS/21

SUBMITTED TO :- MRS. PARUL CHACHRA MA’AM

1.Write a program (using fork() and/or exec() commands) where parent and child execute:

a) same program, same code.

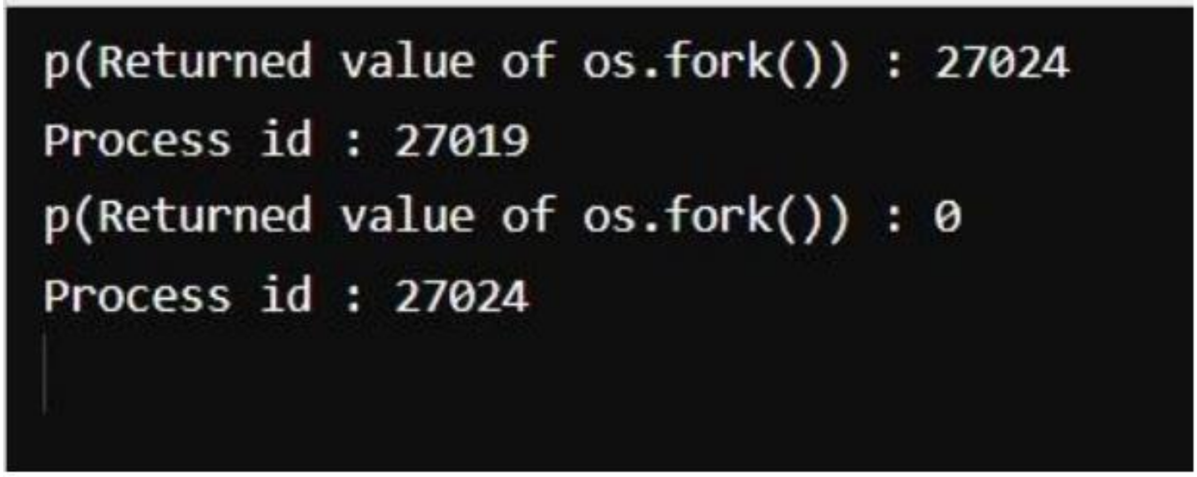
b) same program, different code.

c) before terminating, the parent waits for the child to finish its task.

Solution 1 a)

```
import os
pid = os.fork()
if pid < 0:
 print("Fork failed")
quit()
print(f"p(Returned value of os.fork()) : {pid}")
print(f"Process id : {os.getpid()}")
```

output

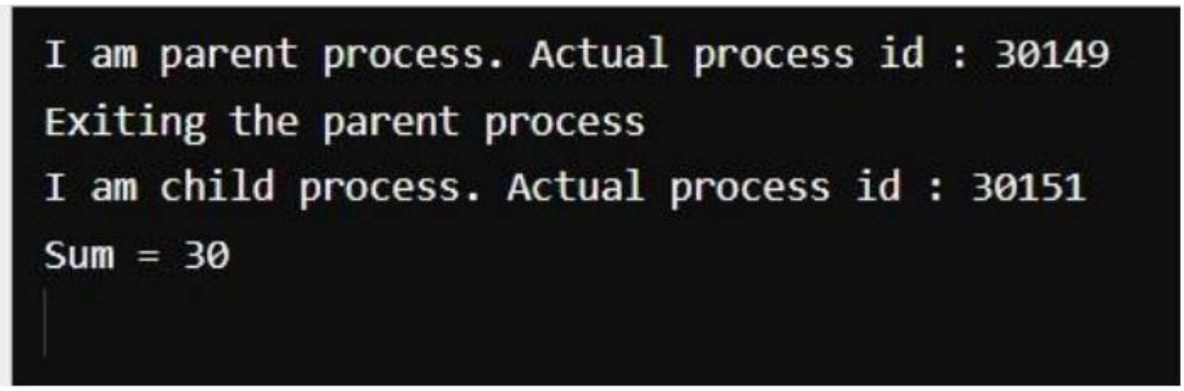


```
p(Returned value of os.fork()) : 27024
Process id : 27019
p(Returned value of os.fork()) : 0
Process id : 27024
```

1 b).

```
import os
pid = os.fork()
# p > 0 ---> Parent process
if pid > 0:
print(f"I am parent process. Actual process id : {os.getpid()} ")
print("Exiting the parent process")
# p == 0 ---> Child process
elif pid == 0:
print(f"I am child process. Actual process id : {os.getpid()}")
newCode = 'a = 10\nb=20\nprint("Sum =", a+b)'
exec(newCode)
else:
print("Forking Error")
quit()
```

output:

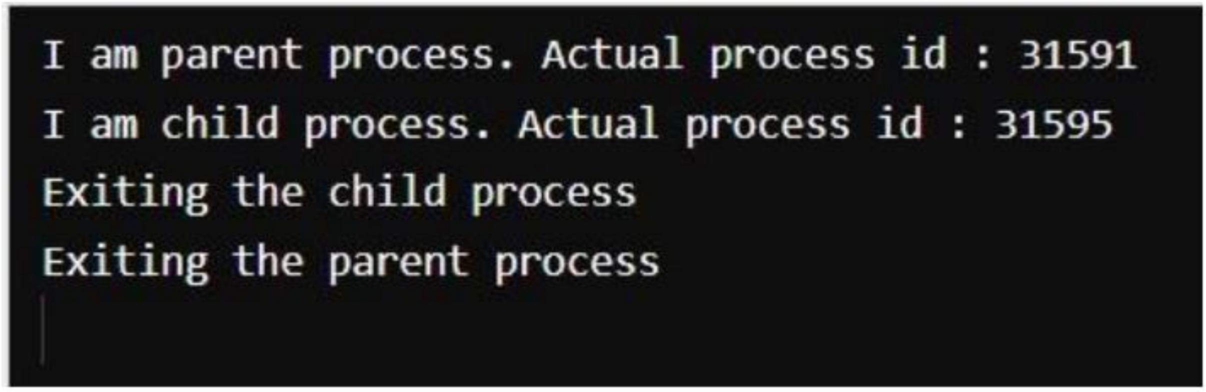


```
I am parent process. Actual process id : 30149
Exiting the parent process
I am child process. Actual process id : 30151
Sum = 30
```

1 c).

```
import os
pid = os.fork()
if pid > 0:
print(f"I am parent process. Actual process id : {os.getpid()} ")
os.waitpid(-1, 0)
print("Exiting the parent process")
elif pid == 0:
print(f"I am child process. Actual process id : {os.getpid()}")
print("Exiting the child process")
else:
print("Forking Error")
quit()
```

output:

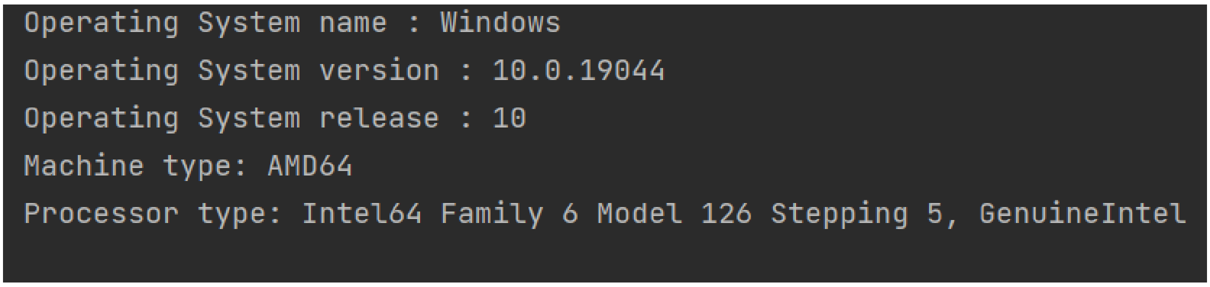


```
I am parent process. Actual process id : 31591
I am child process. Actual process id : 31595
Exiting the child process
Exiting the parent process
```

2. Write a program to report behaviour of Linux kernel including kernel version, CPU type and model. (CPU information).

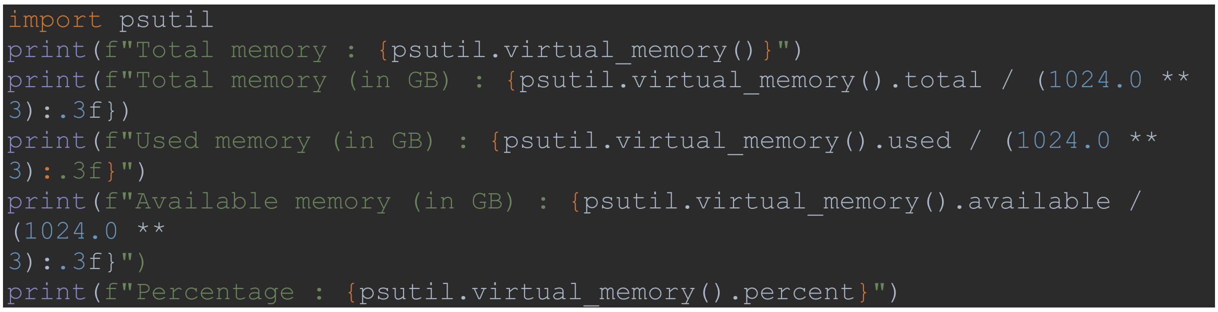
```
import platform
print(f"Operating System name : {platform.system()}")
print(f"Operating System version : {platform.version()}")
print(f"Operating System release : {platform.release()}")
print(f"Machine type: {platform.machine()}")
print(f"Processor type: {platform.processor()}")
```

output:



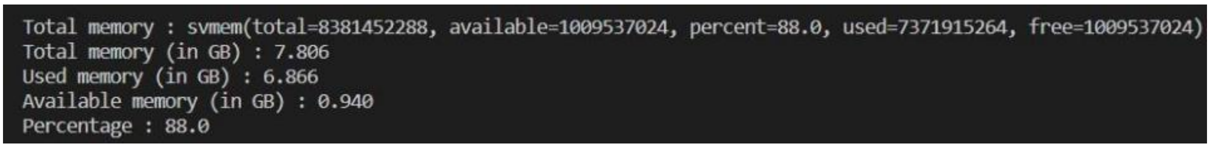
```
Operating System name : Windows
Operating System version : 10.0.19044
Operating System release : 10
Machine type: AMD64
Processor type: Intel64 Family 6 Model 126 Stepping 5, GenuineIntel
```

3.Write a program to report behaviour of Linux kernel including information on 19 configured memory, amount of free and used memory. (memory information)



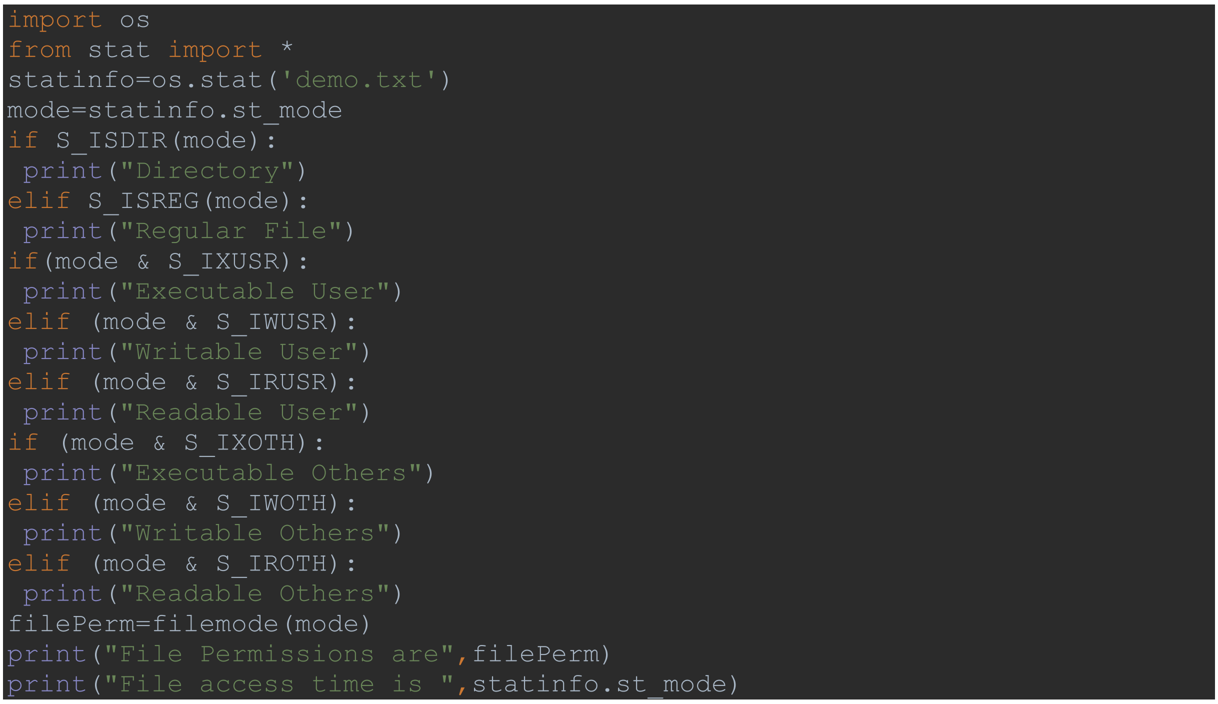
```
import psutil
print(f"Total memory : {psutil.virtual_memory()}")
print(f"Total memory (in GB) : {psutil.virtual_memory().total / (1024.0 **
3):.3f})
print(f"Used memory (in GB) : {psutil.virtual_memory().used / (1024.0 **
3):.3f}")
print(f"Available memory (in GB) : {psutil.virtual_memory().available /
(1024.0 **
3):.3f}")
print(f"Percentage : {psutil.virtual_memory().percent}")
```

output:



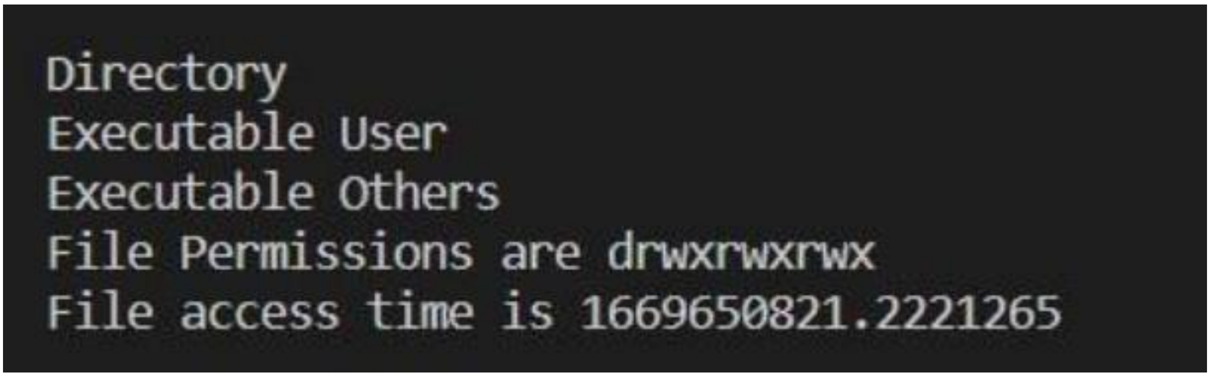
```
Total memory : svmem(total=8381452288, available=1009537024, percent=88.0, used=7371915264, free=1009537024)
Total memory (in GB) : 7.806
Used memory (in GB) : 6.866
Available memory (in GB) : 0.940
Percentage : 88.0
```

4. Write a program to print file details including owner access permissions, file access time, where file name is given as argument.



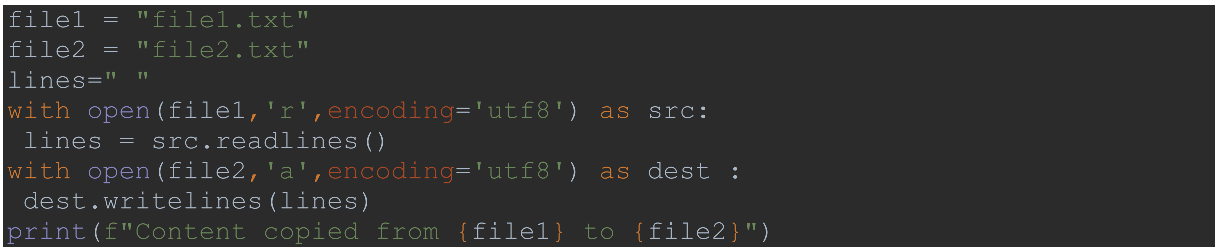
```
import os
from stat import *
statinfo=os.stat('demo.txt')
mode=statinfo.st_mode
if S_ISDIR(mode):
 print("Directory")
elif S_ISREG(mode):
 print("Regular File")
if(mode & S_IXUSR):
 print("Executable User")
elif (mode & S_IWUSR):
 print("Writable User")
elif (mode & S_IRUSR):
 print("Readable User")
if (mode & S_IXOTH):
 print("Executable Others")
elif (mode & S_IWOTH):
 print("Writable Others")
elif (mode & S_IROTH):
 print("Readable Others")
filePerm=filemode(mode)
print("File Permissions are",filePerm)
print("File access time is ",statinfo.st_mode)
```

output:



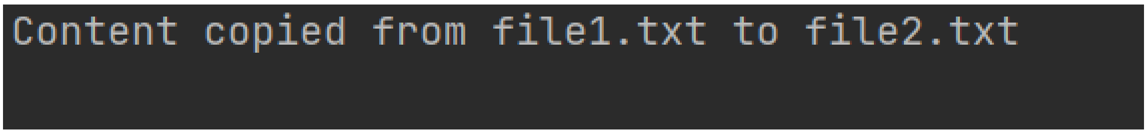
```
Directory
Executable User
Executable Others
File Permissions are drwxrwxrwx
File access time is 1669650821.2221265
```

5. Write a program to copy files using system calls.



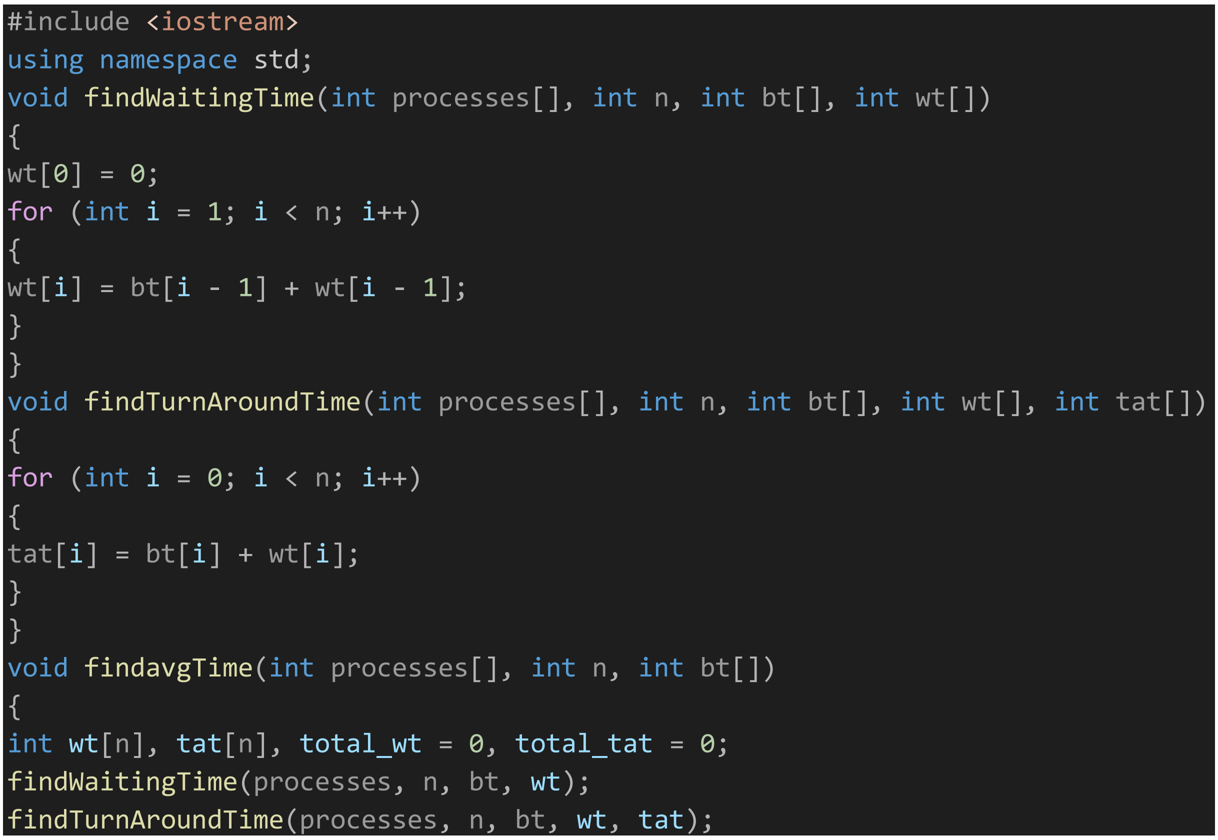
```python
file1 = "file1.txt"
file2 = "file2.txt"
lines=" "
with open(file1,'r',encoding='utf8') as src:
 lines = src.readlines()
with open(file2,'a',encoding='utf8') as dest :
 dest.writelines(lines)
print(f"Content copied from {file1} to {file2}")
```

Output:

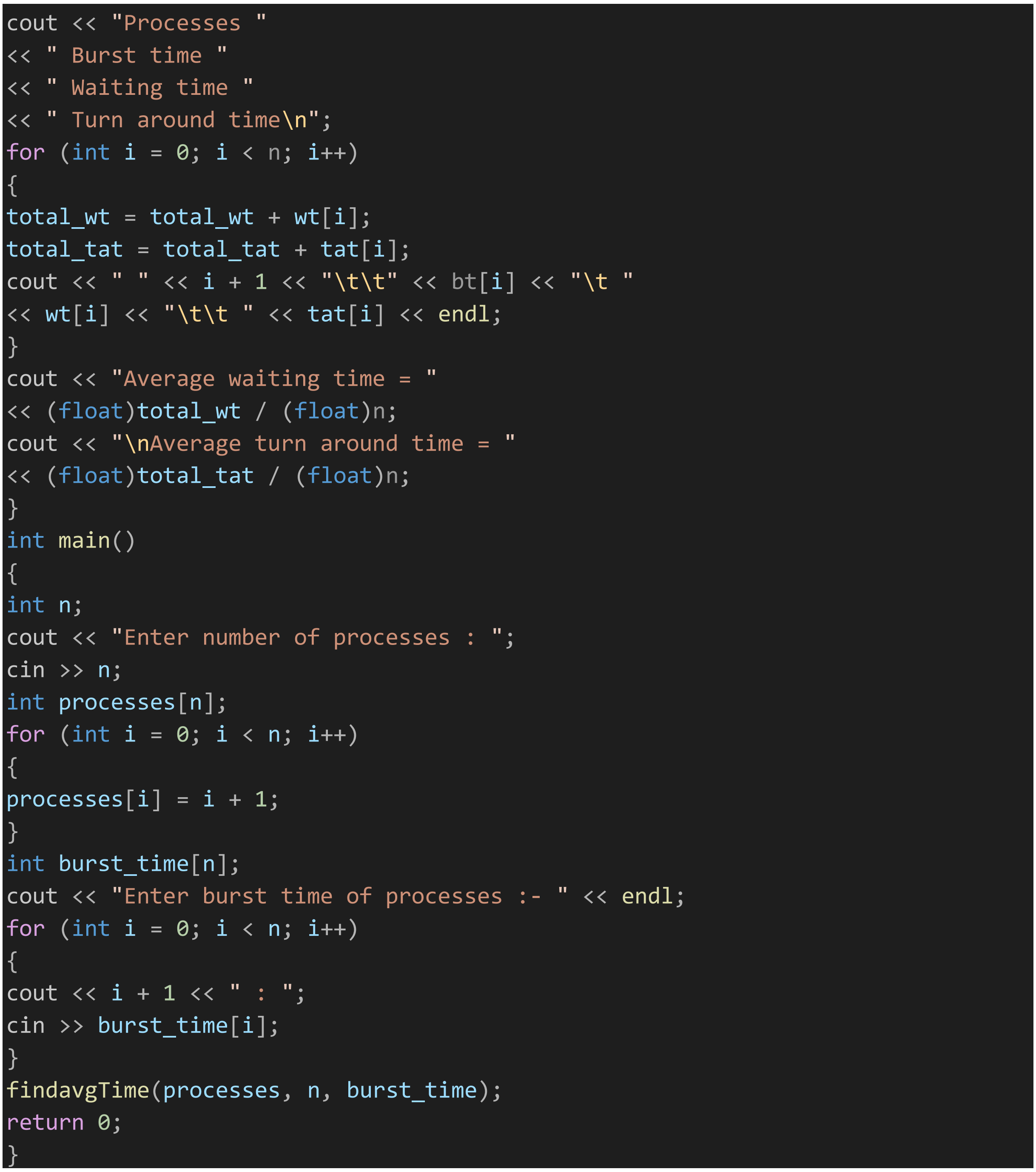


```
Content copied from file1.txt to file2.txt
```

6. Write a program to implement FCFS scheduling algorithm.

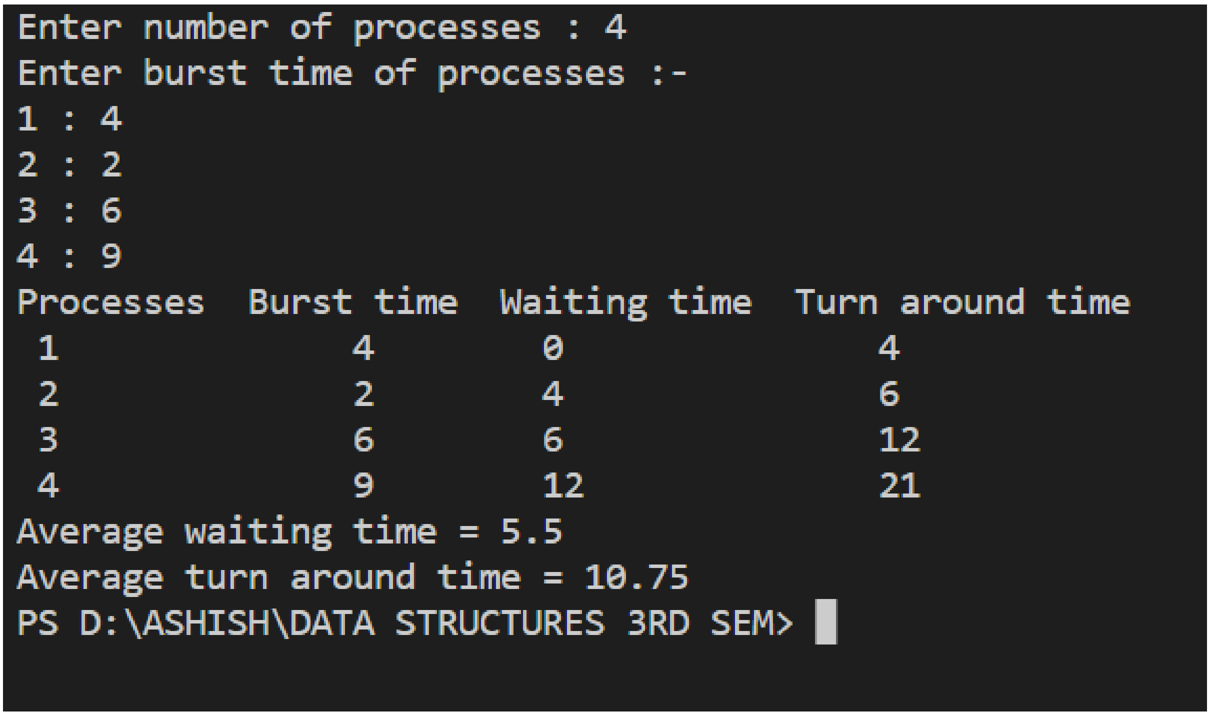


```cpp
#include <iostream>
using namespace std;
void findWaitingTime(int processes[], int n, int bt[], int wt[])
{
wt[0] = 0;
for (int i = 1; i < n; i++)
{
wt[i] = bt[i - 1] + wt[i - 1];
}
}
void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
{
for (int i = 0; i < n; i++)
{
tat[i] = bt[i] + wt[i];
}
}
void findavgTime(int processes[], int n, int bt[])
{
int wt[n], tat[n], total_wt = 0, total_tat = 0;
findWaitingTime(processes, n, bt, wt);
findTurnAroundTime(processes, n, bt, wt, tat);
```



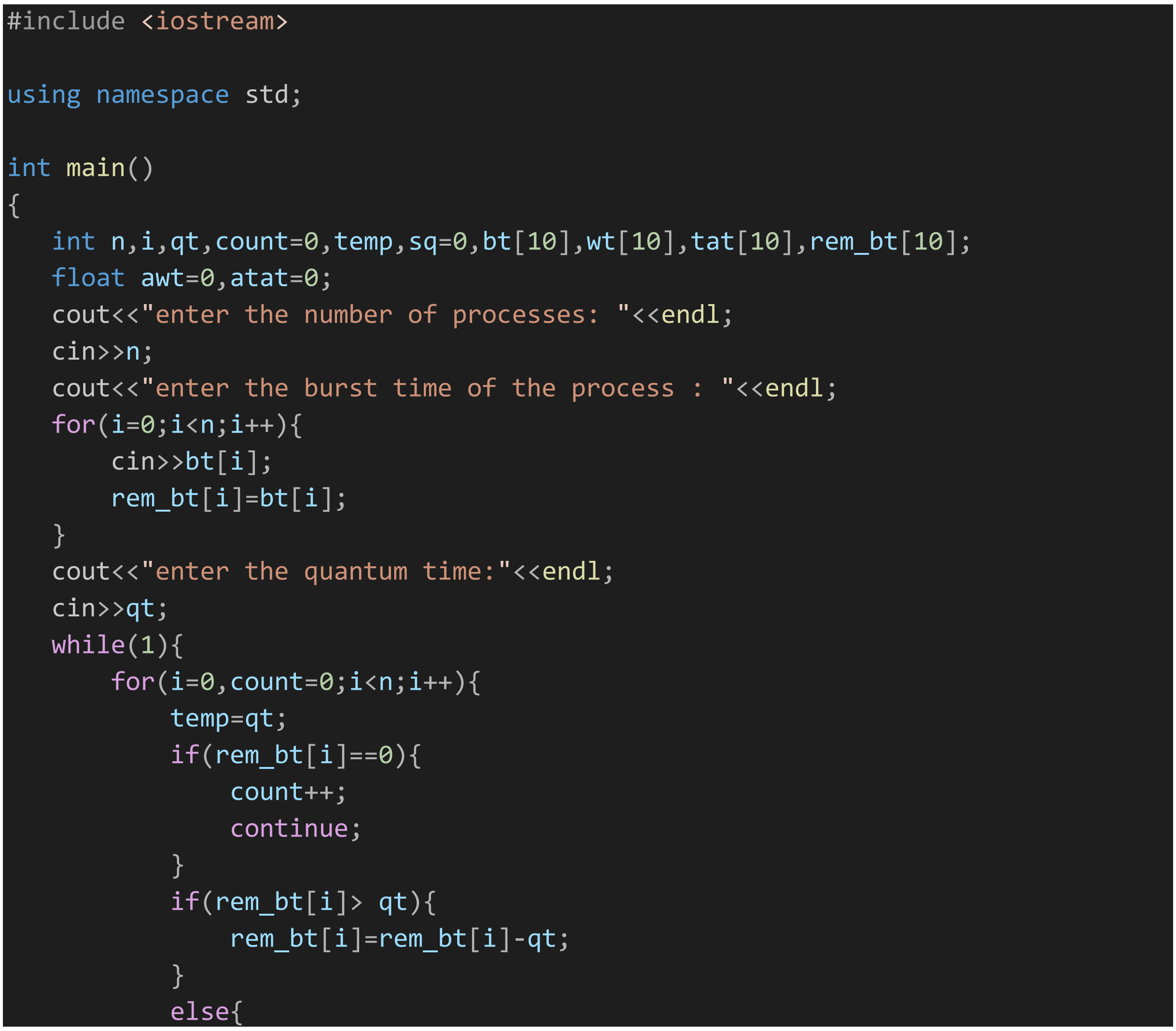
```
cout << "Processes "
<< " Burst time "
<< " Waiting time "
<< " Turn around time\n";
for (int i = 0; i < n; i++)
{
total_wt = total_wt + wt[i];
total_tat = total_tat + tat[i];
cout << " " << i + 1 << "\t\t" << bt[i] << "\t "
<< wt[i] << "\t\t " << tat[i] << endl;
}
cout << "Average waiting time = "
<< (float)total_wt / (float)n;
cout << "\nAverage turn around time = "
<< (float)total_tat / (float)n;
}
int main()
{
int n;
cout << "Enter number of processes : ";
cin >> n;
int processes[n];
for (int i = 0; i < n; i++)
{
processes[i] = i + 1;
}
int burst_time[n];
cout << "Enter burst time of processes :- " << endl;
for (int i = 0; i < n; i++)
{
cout << i + 1 << " : ";
cin >> burst_time[i];
}
findavgTime(processes, n, burst_time);
return 0;
}
```

Output:



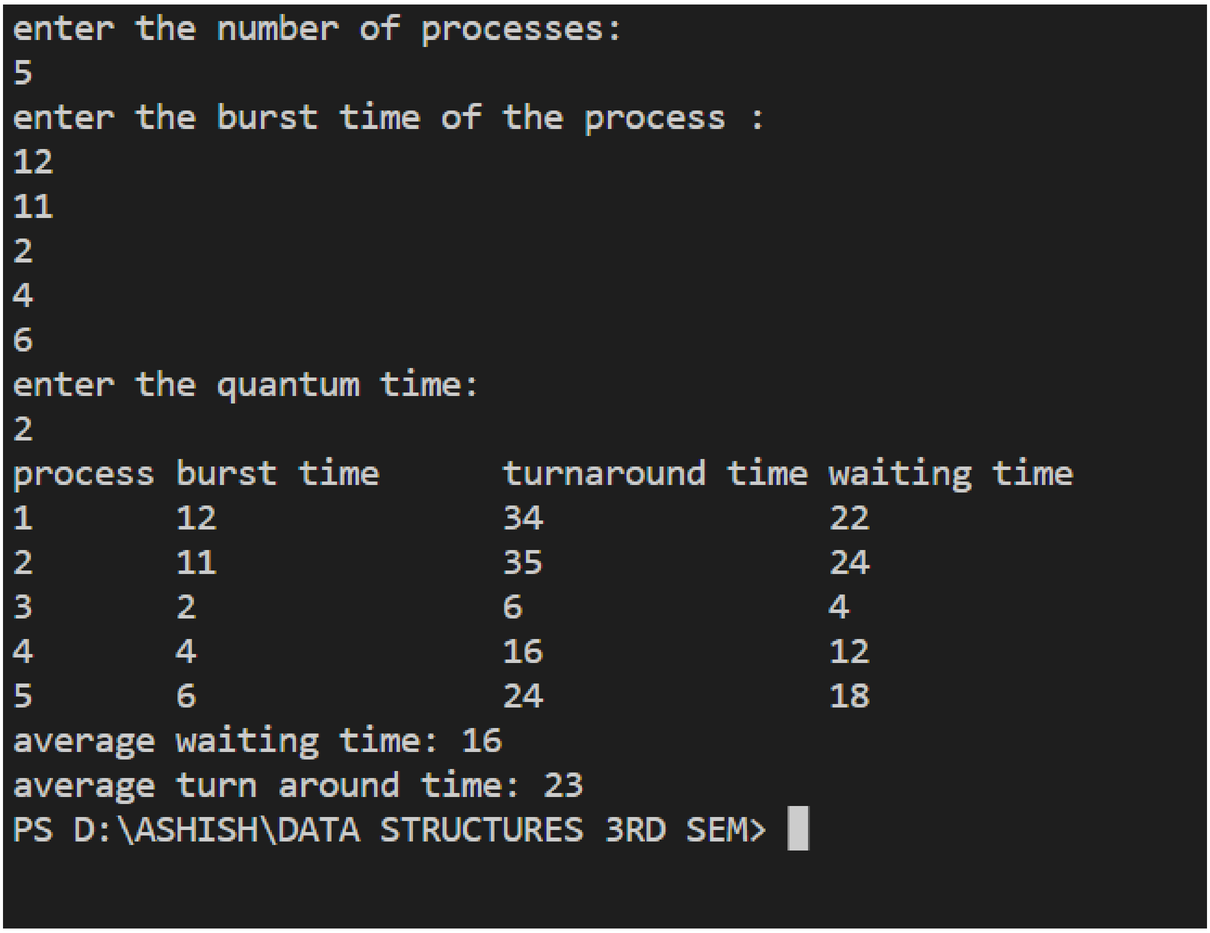
```
Enter number of processes : 4
Enter burst time of processes :-
1 : 4
2 : 2
3 : 6
4 : 9
Processes  Burst time  Waiting time  Turn around time
 1              4        0               4
 2              2        4               6
 3              6        6               12
 4              9        12              21
Average waiting time = 5.5
Average turn around time = 10.75
PS D:\ASHISH\DATA STRUCTURES 3RD SEM> 
```

7. Write a program to implement Round Robin scheduling algorithm.



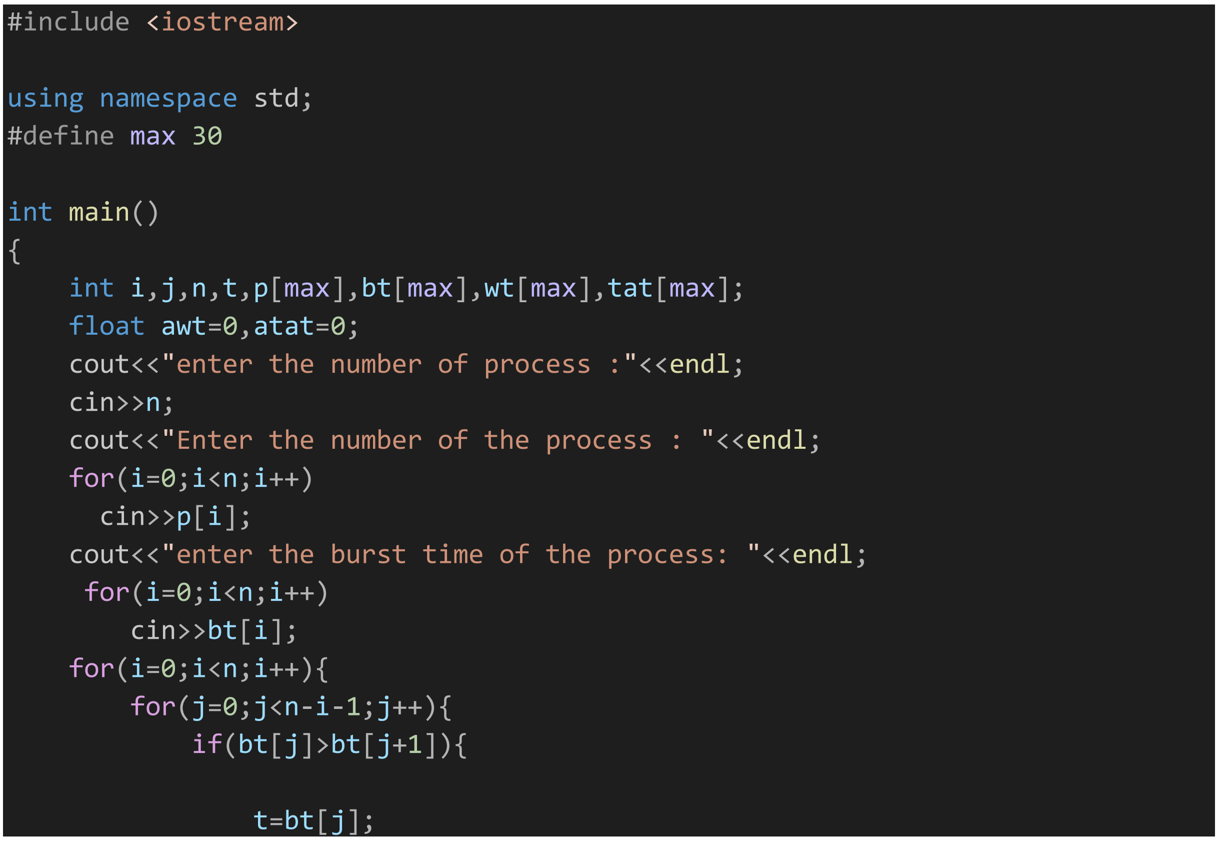
```
#include <iostream>

using namespace std;

int main()
{
   int n,i,qt,count=0,temp,sq=0,bt[10],wt[10],tat[10],rem_bt[10];
   float awt=0,atat=0;
   cout<<"enter the number of processes: "<<endl;
   cin>>n;
   cout<<"enter the burst time of the process : "<<endl;
   for(i=0;i<n;i++){
       cin>>bt[i];
       rem_bt[i]=bt[i];
   }
   cout<<"enter the quantum time:"<<endl;
   cin>>qt;
   while(1){
       for(i=0,count=0;i<n;i++){
           temp=qt;
           if(rem_bt[i]==0){
               count++;
               continue;
           }
           if(rem_bt[i]> qt){
               rem_bt[i]=rem_bt[i]-qt;
           }
           else{
               if(rem_bt[i]>=0){
                   temp=rem_bt[i];
                   rem_bt[i]=0;
               }
           }
           sq=sq+temp;
           tat[i]=sq;
       }
       if(n==count){
           break;
       }
   }
   cout<<"process"<<"\t"<<"burst time"<<"\t"<<"turnaround
time"<<"\t"<<"waiting time"<<endl;
   for(i=0;i<n;i++){
       wt[i]=tat[i]-bt[i];
       awt=awt+wt[i];
       atat=atat+tat[i];
       cout<<i+1<<"\t"<<bt[i]<<"\t\t"<<tat[i]<<"\t\t"<<wt[i]<<endl;
       }
       awt=awt/n;
       atat=atat/n;
    cout<<"average waiting time: "<<awt<<endl;
     cout<<"average turn around time: "<<atat<<endl;
    return 0;
}
```

Output:

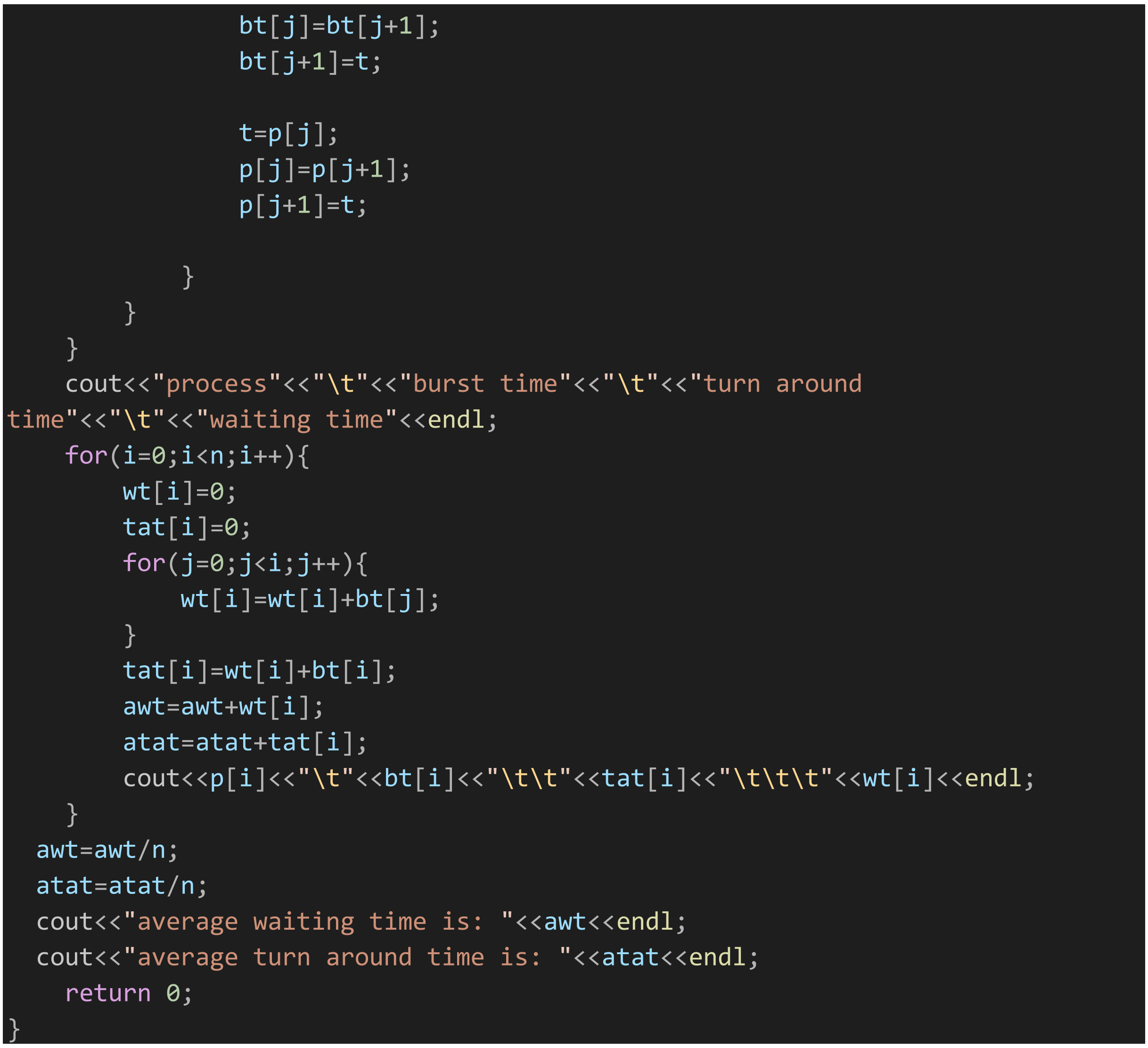


```
enter the number of processes:
5
enter the burst time of the process :
12
11
2
4
6
enter the quantum time:
2
process burst time      turnaround time waiting time
1       12              34              22
2       11              35              24
3       2               6               4
4       4               16              12
5       6               24              18
average waiting time: 16
average turn around time: 23
PS D:\ASHISH\DATA STRUCTURES 3RD SEM> 
```

8. Write a program to implement SJF scheduling algorithm.

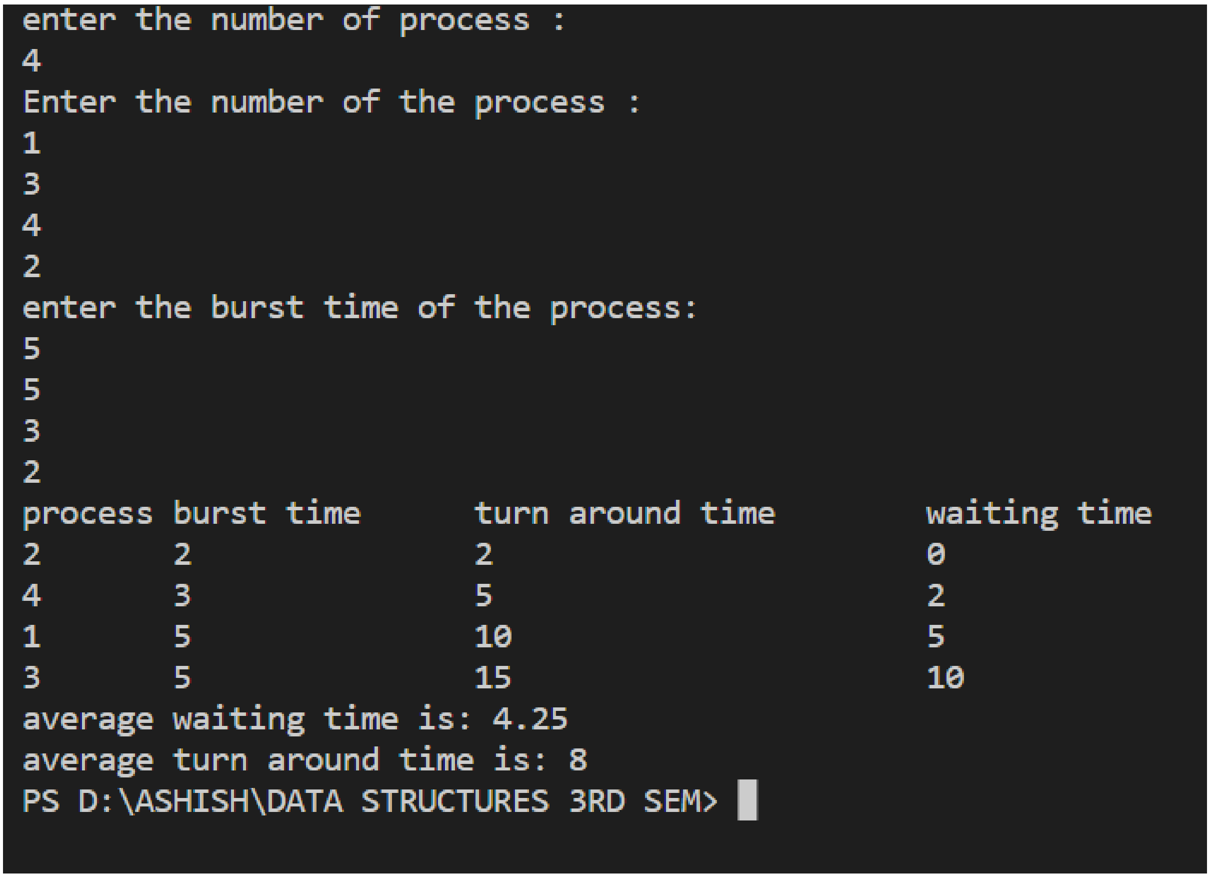


```
#include <iostream>

using namespace std;
#define max 30

int main()
{
    int i,j,n,t,p[max],bt[max],wt[max],tat[max];
    float awt=0,atat=0;
    cout<<"enter the number of process :"<<endl;
    cin>>n;
    cout<<"Enter the number of the process : "<<endl;
    for(i=0;i<n;i++)
      cin>>p[i];
    cout<<"enter the burst time of the process: "<<endl;
     for(i=0;i<n;i++)
        cin>>bt[i];
    for(i=0;i<n;i++){
        for(j=0;j<n-i-1;j++){
            if(bt[j]>bt[j+1]){

                t=bt[j];
```



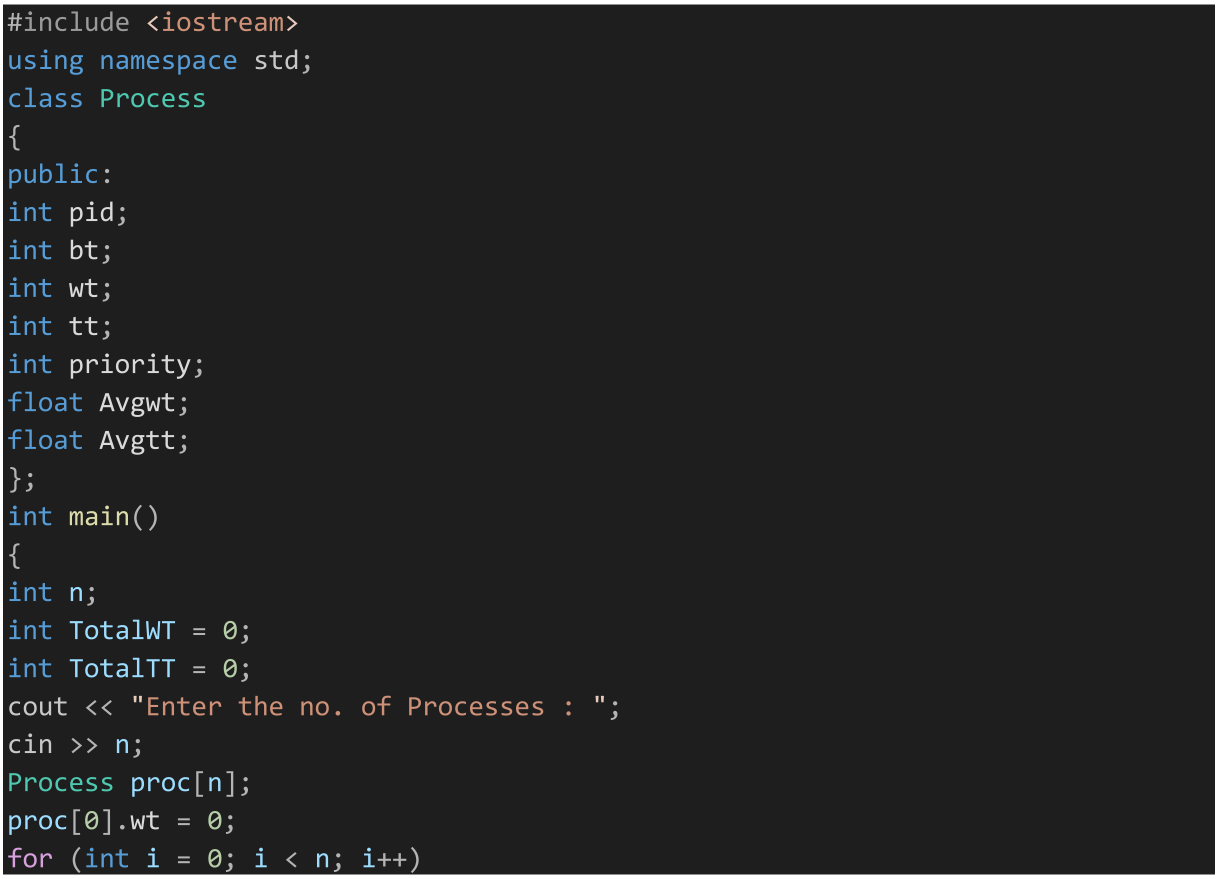
```
                bt[j]=bt[j+1];
                bt[j+1]=t;

                t=p[j];
                p[j]=p[j+1];
                p[j+1]=t;

            }
        }
    }
    cout<<"process"<<"\t"<<"burst time"<<"\t"<<"turn around
time"<<"\t"<<"waiting time"<<endl;
    for(i=0;i<n;i++){
        wt[i]=0;
        tat[i]=0;
        for(j=0;j<i;j++){
            wt[i]=wt[i]+bt[j];
        }
        tat[i]=wt[i]+bt[i];
        awt=awt+wt[i];
        atat=atat+tat[i];
        cout<<p[i]<<"\t"<<bt[i]<<"\t\t"<<tat[i]<<"\t\t\t"<<wt[i]<<endl;
    }
  awt=awt/n;
  atat=atat/n;
  cout<<"average waiting time is: "<<awt<<endl;
  cout<<"average turn around time is: "<<atat<<endl;
    return 0;
}
```

Output:

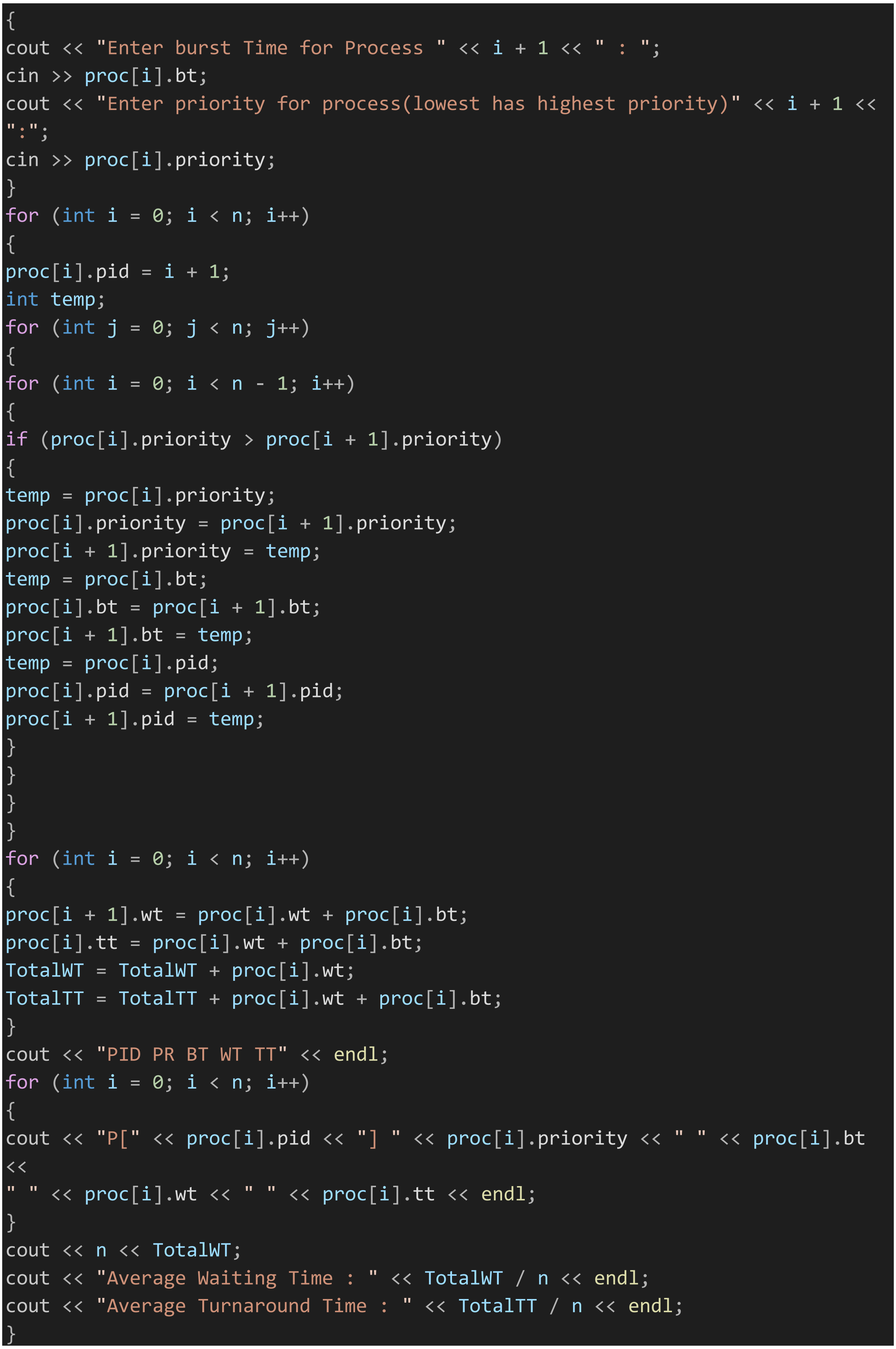


```
enter the number of process :
4
Enter the number of the process :
1
3
4
2
enter the burst time of the process:
5
5
3
2
process burst time      turn around time        waiting time
2       2               2                       0
4       3               5                       2
1       5               10                      5
3       5               15                      10
average waiting time is: 4.25
average turn around time is: 8
PS D:\ASHISH\DATA STRUCTURES 3RD SEM> 
```

9. Write a program to implement non-preemptive priority based scheduling algorithm.

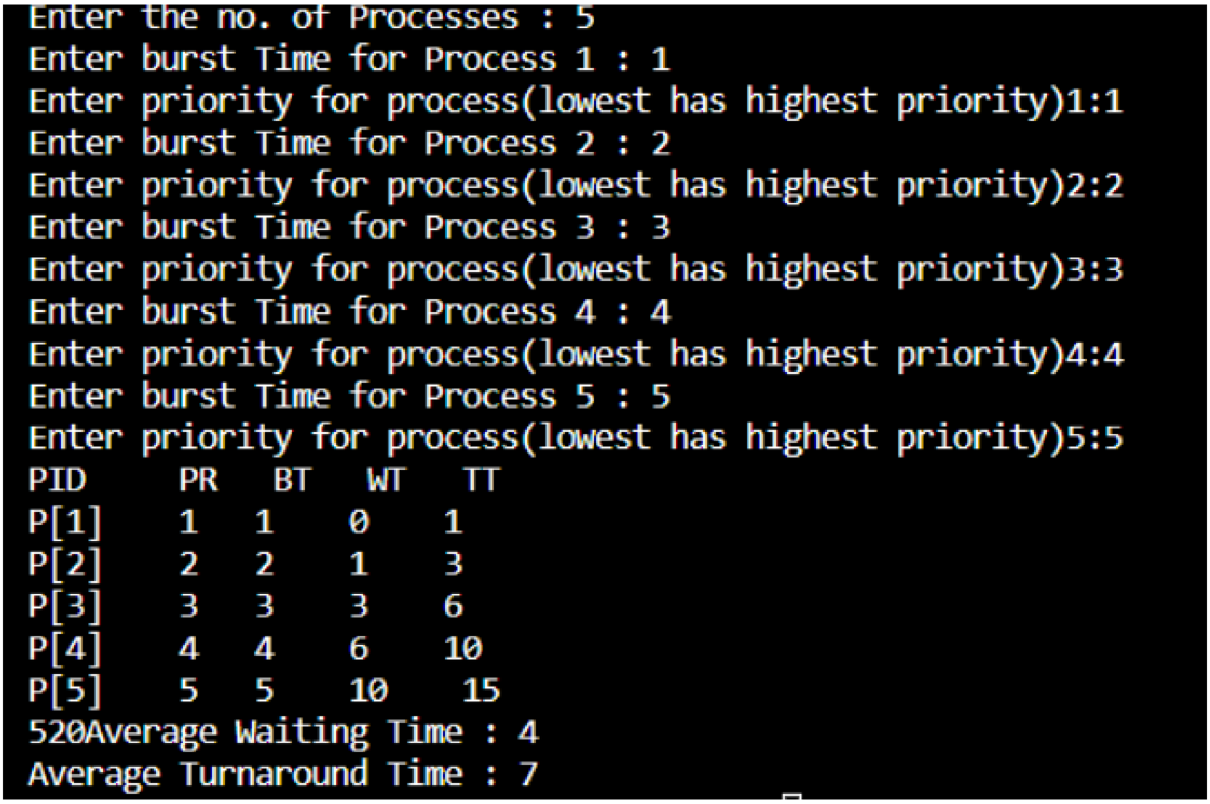


```
#include <iostream>
using namespace std;
class Process
{
public:
int pid;
int bt;
int wt;
int tt;
int priority;
float Avgwt;
float Avgtt;
};
int main()
{
int n;
int TotalWT = 0;
int TotalTT = 0;
cout << "Enter the no. of Processes : ";
cin >> n;
Process proc[n];
proc[0].wt = 0;
for (int i = 0; i < n; i++)
```



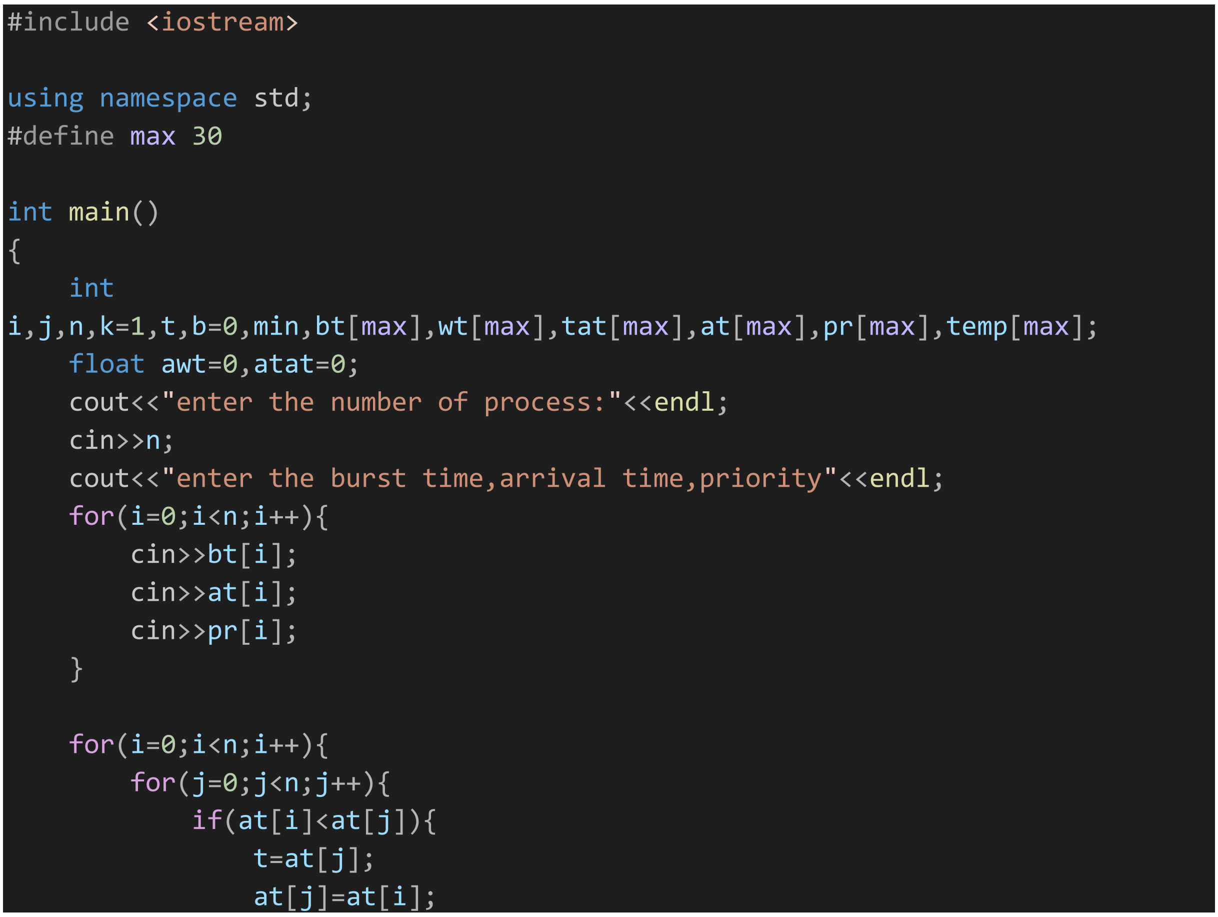
```
{
cout << "Enter burst Time for Process " << i + 1 << " : ";
cin >> proc[i].bt;
cout << "Enter priority for process(lowest has highest priority)" << i + 1 <<
":";
cin >> proc[i].priority;
}
for (int i = 0; i < n; i++)
{
proc[i].pid = i + 1;
int temp;
for (int j = 0; j < n; j++)
{
for (int i = 0; i < n - 1; i++)
{
if (proc[i].priority > proc[i + 1].priority)
{
temp = proc[i].priority;
proc[i].priority = proc[i + 1].priority;
proc[i + 1].priority = temp;
temp = proc[i].bt;
proc[i].bt = proc[i + 1].bt;
proc[i + 1].bt = temp;
temp = proc[i].pid;
proc[i].pid = proc[i + 1].pid;
proc[i + 1].pid = temp;
}
}
}
}
for (int i = 0; i < n; i++)
{
proc[i + 1].wt = proc[i].wt + proc[i].bt;
proc[i].tt = proc[i].wt + proc[i].bt;
TotalWT = TotalWT + proc[i].wt;
TotalTT = TotalTT + proc[i].wt + proc[i].bt;
}
cout << "PID PR BT WT TT" << endl;
for (int i = 0; i < n; i++)
{
cout << "P[" << proc[i].pid << "] " << proc[i].priority << " " << proc[i].bt
<<
" " << proc[i].wt << " " << proc[i].tt << endl;
}
cout << n << TotalWT;
cout << "Average Waiting Time : " << TotalWT / n << endl;
cout << "Average Turnaround Time : " << TotalTT / n << endl;
}
```

Output:

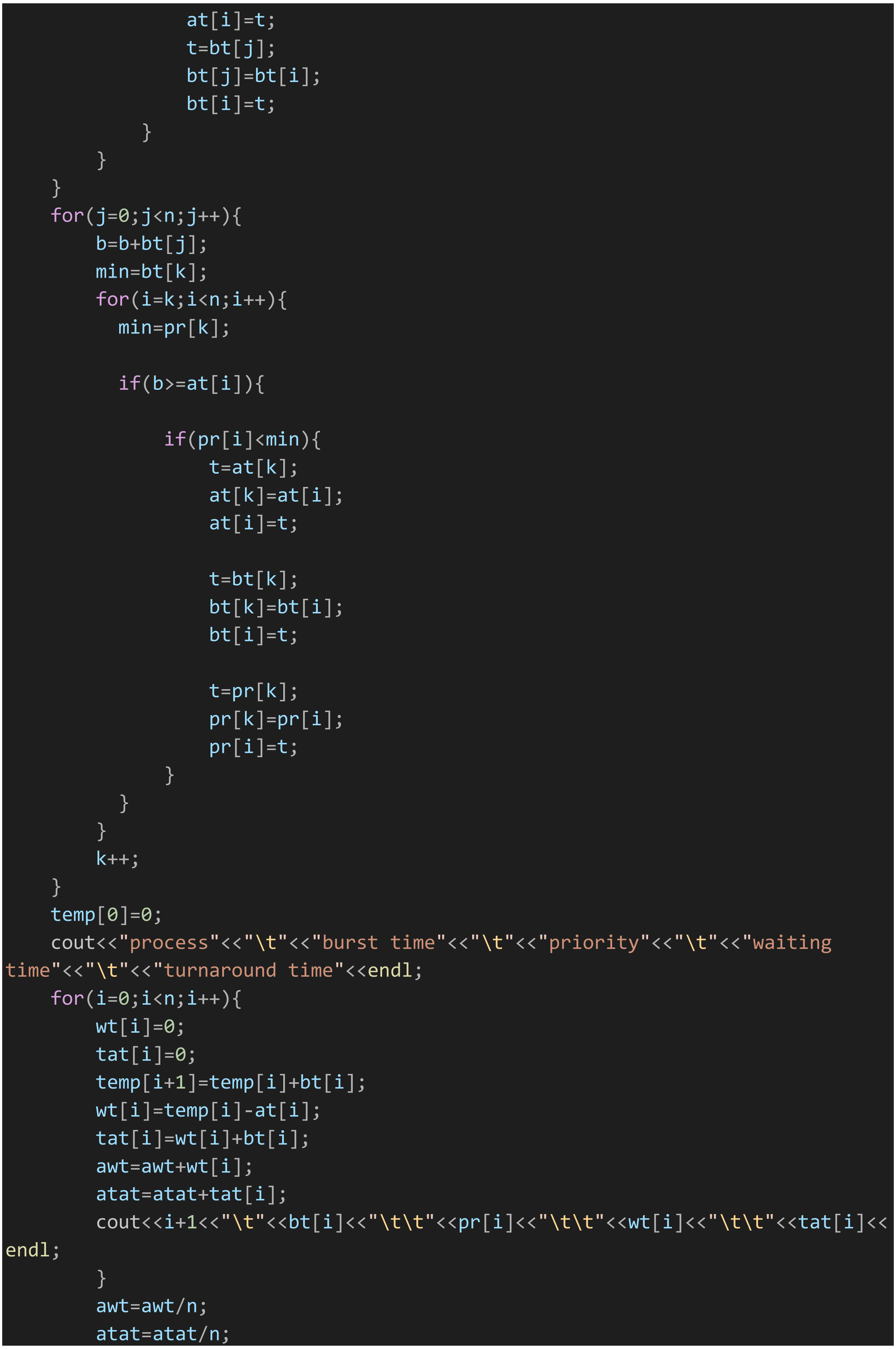


```
Enter the no. of Processes : 5
Enter burst Time for Process 1 : 1
Enter priority for process(lowest has highest priority)1:1
Enter burst Time for Process 2 : 2
Enter priority for process(lowest has highest priority)2:2
Enter burst Time for Process 3 : 3
Enter priority for process(lowest has highest priority)3:3
Enter burst Time for Process 4 : 4
Enter priority for process(lowest has highest priority)4:4
Enter burst Time for Process 5 : 5
Enter priority for process(lowest has highest priority)5:5
PID      PR   BT   WT   TT
P[1]     1    1    0    1
P[2]     2    2    1    3
P[3]     3    3    3    6
P[4]     4    4    6    10
P[5]     5    5    10    15
520Average Waiting Time : 4
Average Turnaround Time : 7
```

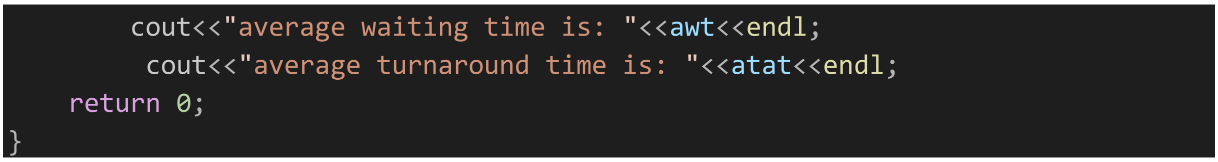
10. Write a program to implement a preemptive priority based scheduling algorithm.



```
#include <iostream>

using namespace std;
#define max 30

int main()
{
    int
i,j,n,k=1,t,b=0,min,bt[max],wt[max],tat[max],at[max],pr[max],temp[max];
    float awt=0,atat=0;
    cout<<"enter the number of process:"<<endl;
    cin>>n;
    cout<<"enter the burst time,arrival time,priority"<<endl;
    for(i=0;i<n;i++){
        cin>>bt[i];
        cin>>at[i];
        cin>>pr[i];
    }

    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            if(at[i]<at[j]){
                t=at[j];
                at[j]=at[i];
```

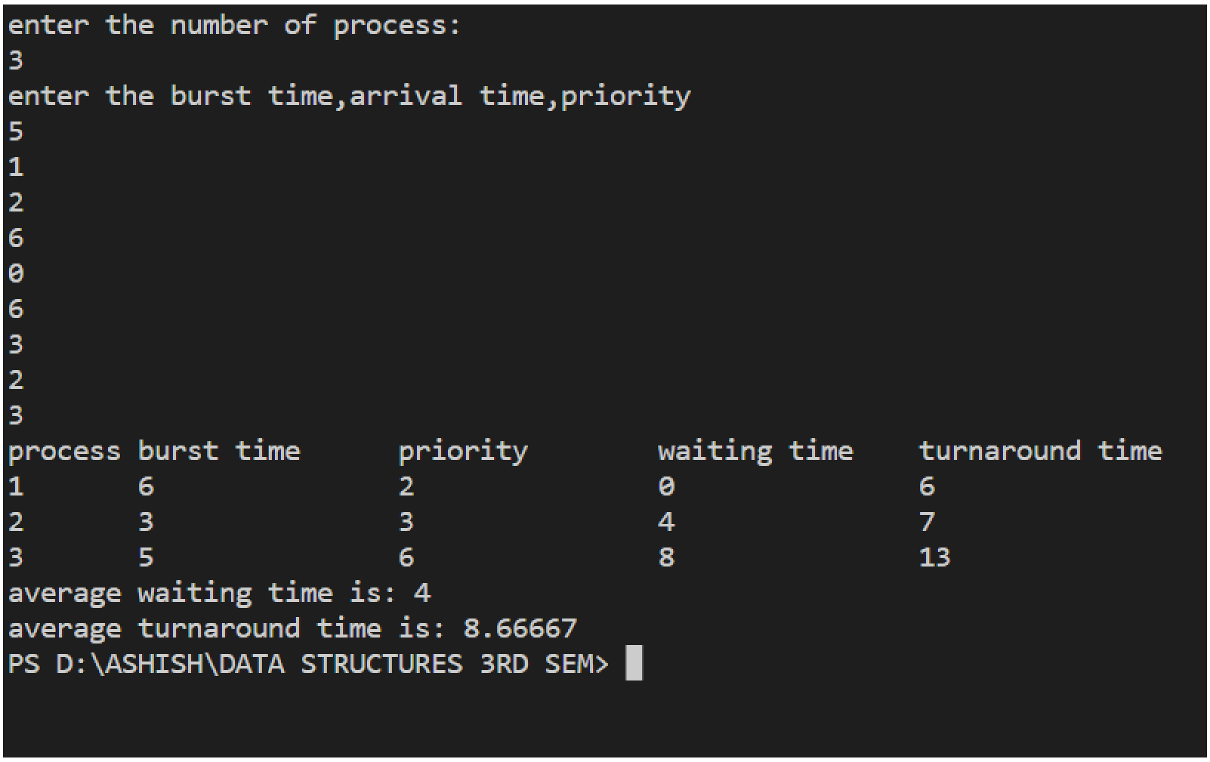


```
        at[i]=t;
        t=bt[j];
        bt[j]=bt[i];
        bt[i]=t;
      }
    }
  }
  for(j=0;j<n;j++){
    b=b+bt[j];
    min=bt[k];
    for(i=k;i<n;i++){
     min=pr[k];

     if(b>=at[i]){

       if(pr[i]<min){
         t=at[k];
         at[k]=at[i];
         at[i]=t;

         t=bt[k];
         bt[k]=bt[i];
         bt[i]=t;

         t=pr[k];
         pr[k]=pr[i];
         pr[i]=t;
       }
     }
    }
    k++;
  }
  temp[0]=0;
  cout<<"process"<<"\t"<<"burst time"<<"\t"<<"priority"<<"\t"<<"waiting
time"<<"\t"<<"turnaround time"<<endl;
  for(i=0;i<n;i++){
    wt[i]=0;
    tat[i]=0;
    temp[i+1]=temp[i]+bt[i];
    wt[i]=temp[i]-at[i];
    tat[i]=wt[i]+bt[i];
    awt=awt+wt[i];
    atat=atat+tat[i];
    cout<<i+1<<"\t"<<bt[i]<<"\t\t"<<pr[i]<<"\t\t"<<wt[i]<<"\t\t"<<tat[i]<<
endl;
    }
    awt=awt/n;
    atat=atat/n;
```



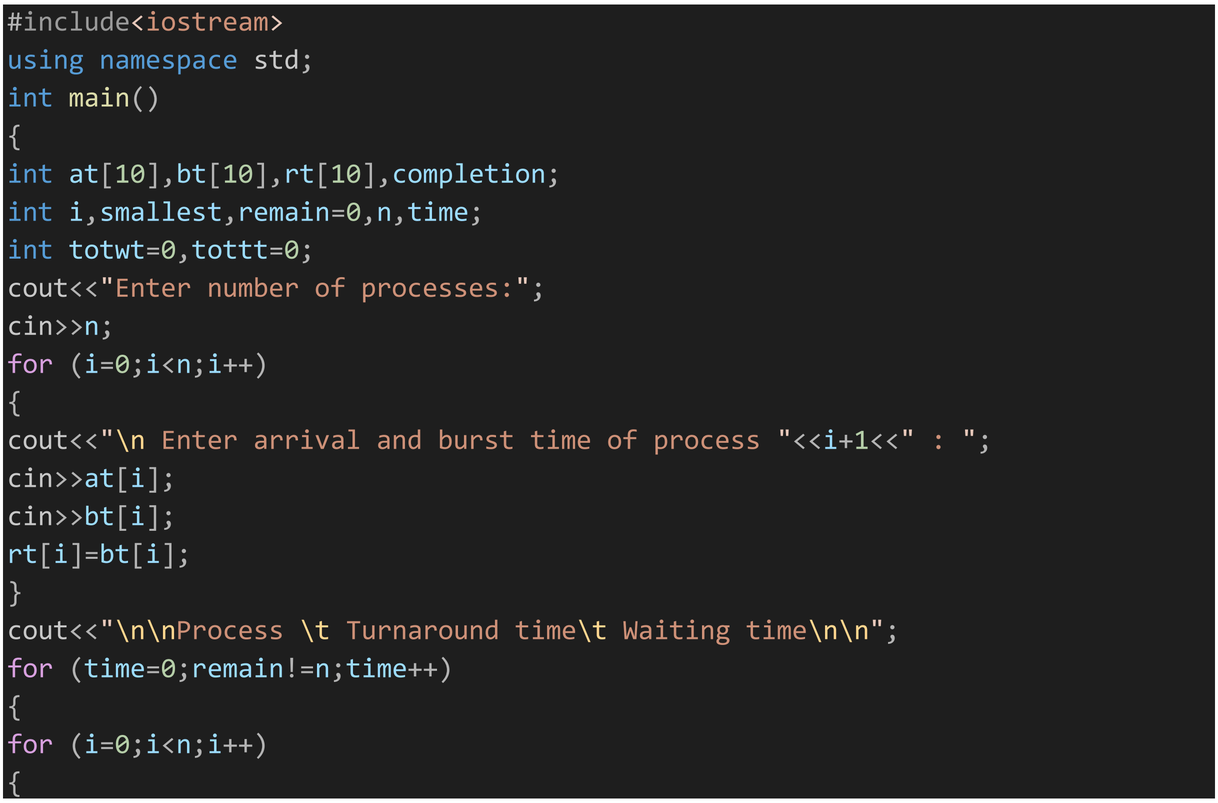
```
        cout<<"average waiting time is: "<<awt<<endl;
         cout<<"average turnaround time is: "<<atat<<endl;
    return 0;
}
```

Output:



```
enter the number of process:
3
enter the burst time,arrival time,priority
5
1
2
6
0
6
3
2
3
process burst time      priority        waiting time    turnaround time
1       6               2               0               6
2       3               3               4               7
3       5               6               8               13
average waiting time is: 4
average turnaround time is: 8.66667
PS D:\ASHISH\DATA STRUCTURES 3RD SEM> █
```

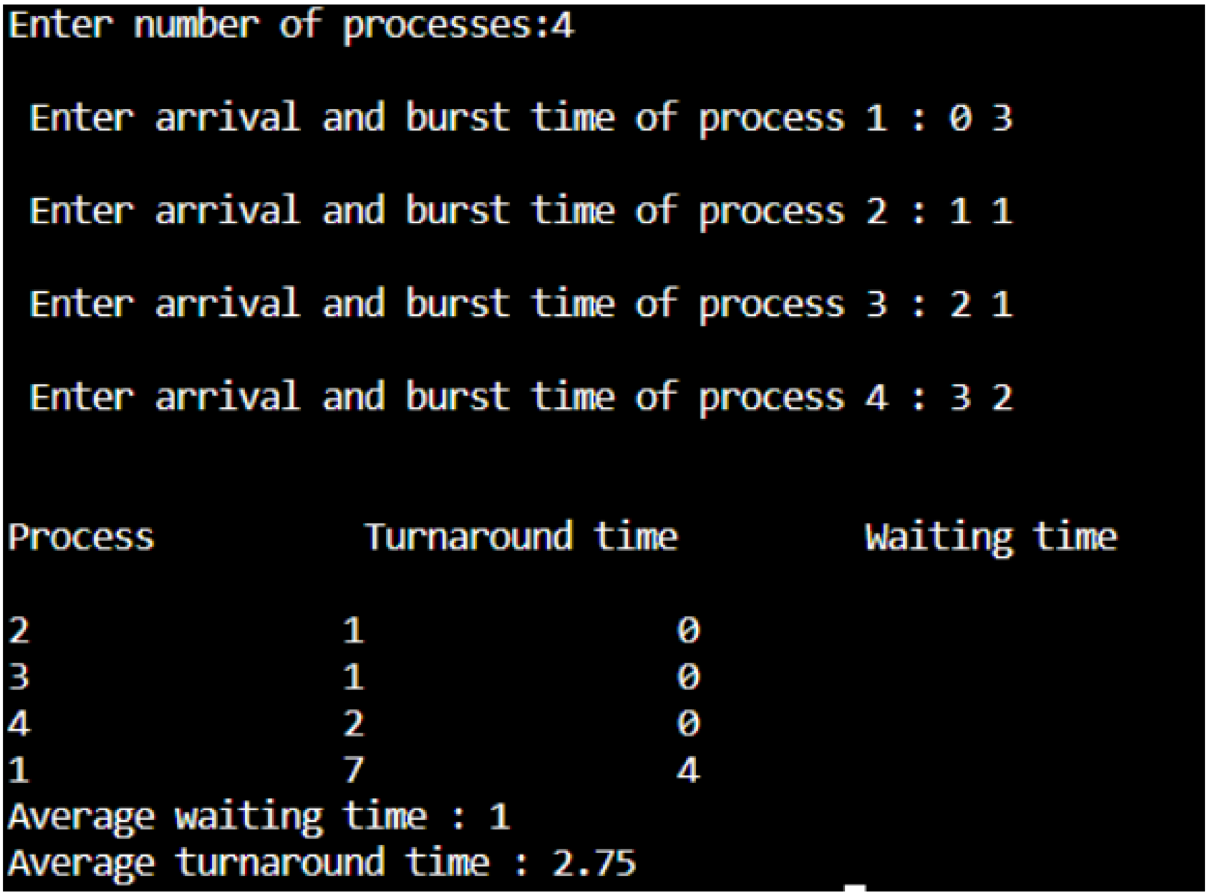
11. Write a program to implement SRJF scheduling algorithm.



```
#include<iostream>
using namespace std;
int main()
{
int at[10],bt[10],rt[10],completion;
int i,smallest,remain=0,n,time;
int totwt=0,tottt=0;
cout<<"Enter number of processes:";
cin>>n;
for (i=0;i<n;i++)
{
cout<<"\n Enter arrival and burst time of process "<<i+1<<" : ";
cin>>at[i];
cin>>bt[i];
rt[i]=bt[i];
}
cout<<"\n\nProcess \t Turnaround time\t Waiting time\n\n";
for (time=0;remain!=n;time++)
{
for (i=0;i<n;i++)
{
```

```
if (at[i]<=time && rt[i])
{
smallest=i;
}
}
rt[smallest]--;
if (rt[smallest]==0)
{
remain++;
completion=time+1;
int a= completion-bt[smallest]-at[smallest];
cout<<smallest+1<<"\t\t"<<completion-at[smallest]<<"\t\t"<<a<<endl;
totwt = totwt+ a;
tottt += completion-at[smallest];
}
}
cout<<"Average waiting time : "<<totwt*1.0/n<<endl;
cout<<"Average turnaround time : "<<tottt*1.0/n<<endl;
}
```

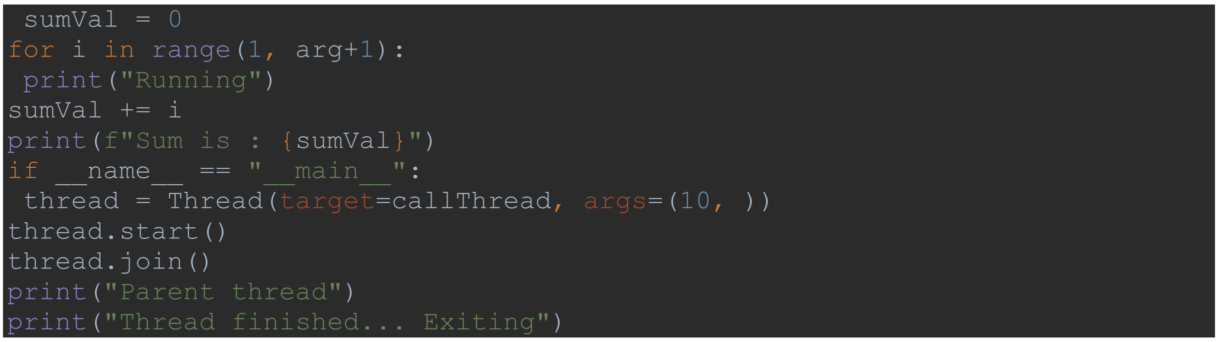
Output:



```
Enter number of processes:4

 Enter arrival and burst time of process 1 : 0 3

 Enter arrival and burst time of process 2 : 1 1

 Enter arrival and burst time of process 3 : 2 1

 Enter arrival and burst time of process 4 : 3 2


Process          Turnaround time          Waiting time

2               1               0
3               1               0
4               2               0
1               7               4
Average waiting time : 1
Average turnaround time : 2.75
```

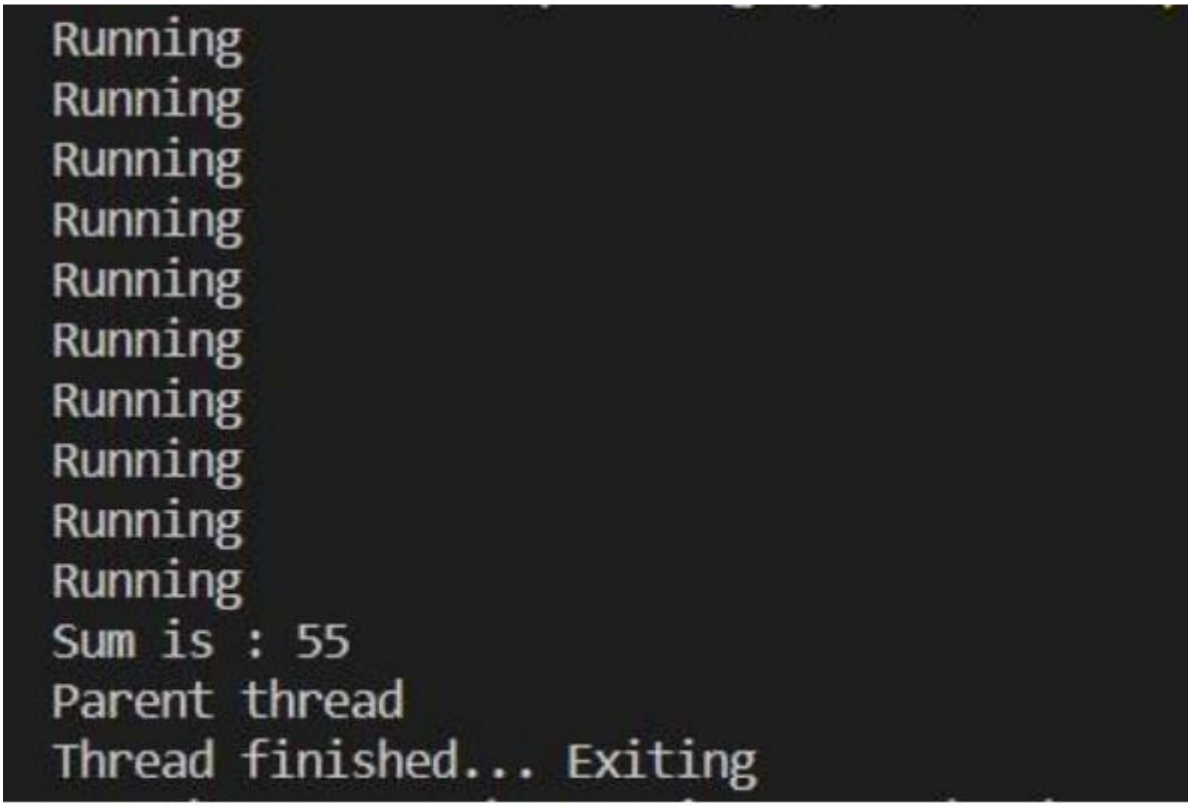
12. Write a program to calculate sum of n numbers using thread library.

```
from threading import Thread
# function to create threads
def callThread(arg):
```



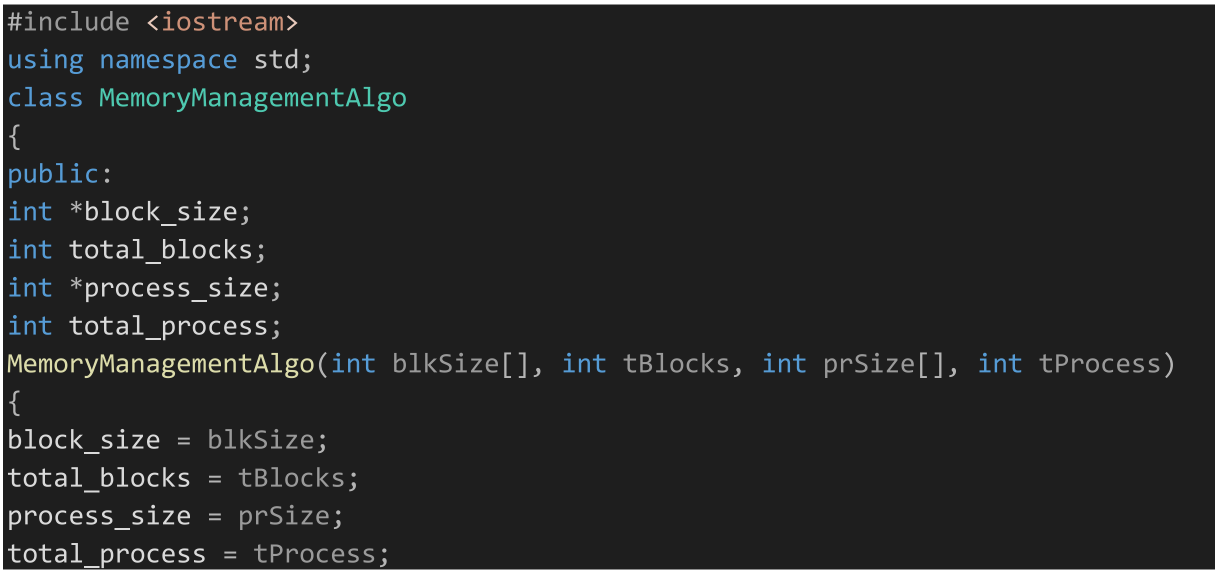
```
 sumVal = 0
for i in range(1, arg+1):
 print("Running")
sumVal += i
print(f"Sum is : {sumVal}")
if __name__ == "__main__":
 thread = Thread(target=callThread, args=(10, ))
thread.start()
thread.join()
print("Parent thread")
print("Thread finished... Exiting")
```

output:

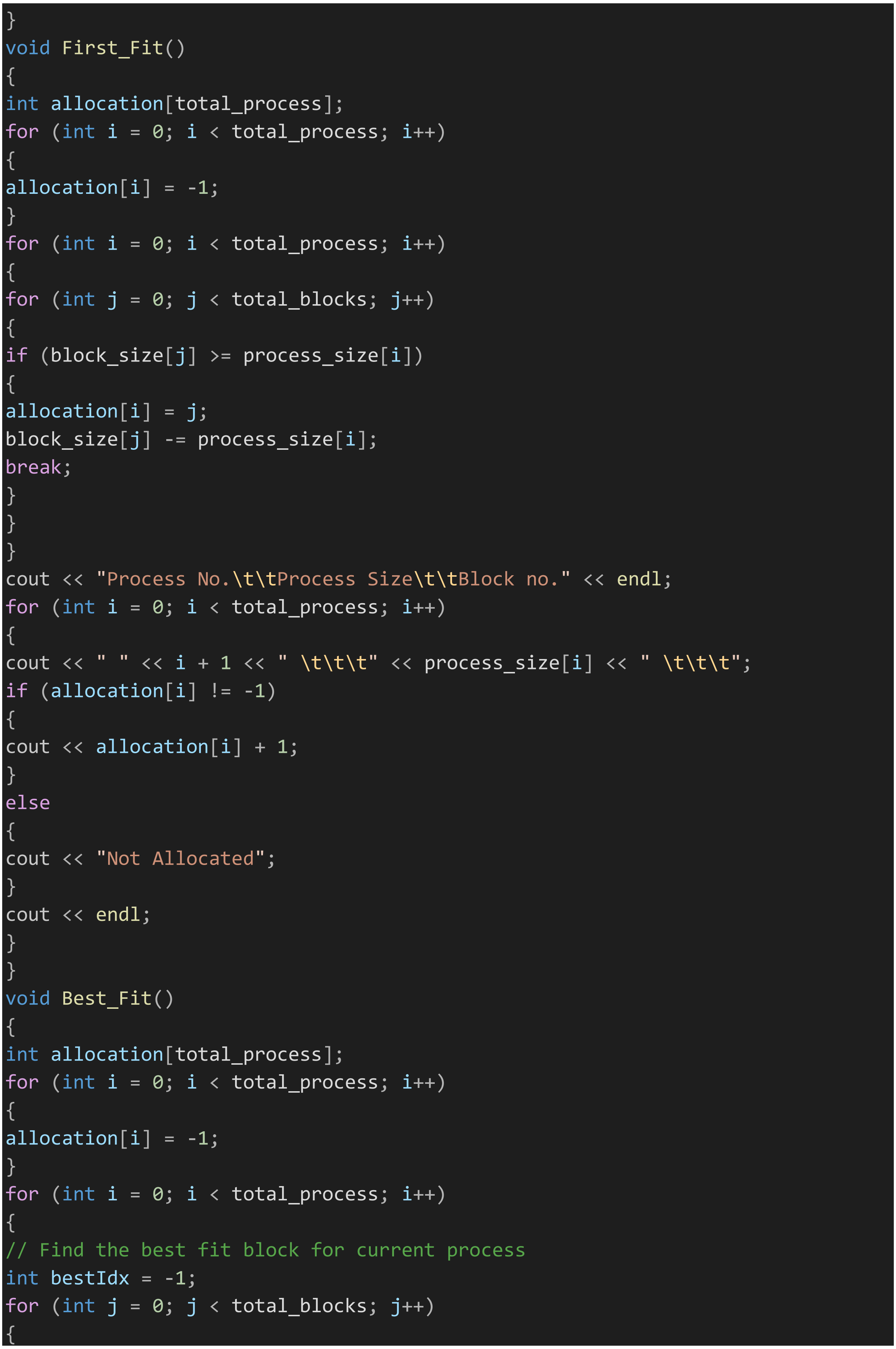


```
Running
Running
Running
Running
Running
Running
Running
Running
Running
Running
Sum is : 55
Parent thread
Thread finished... Exiting
```

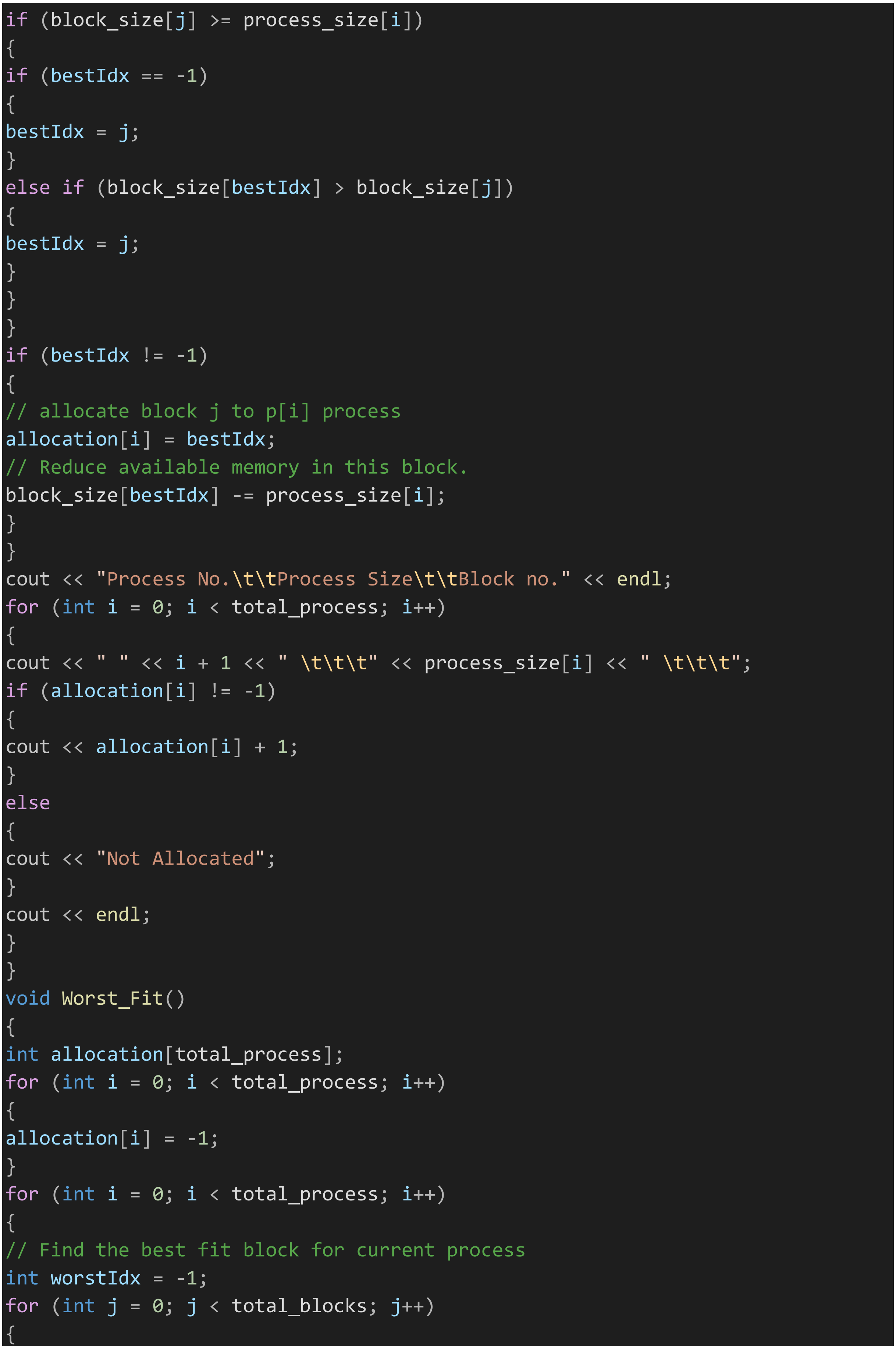
13. Write a program to implement first-fit, best-fit and worst-fit allocation strategies.



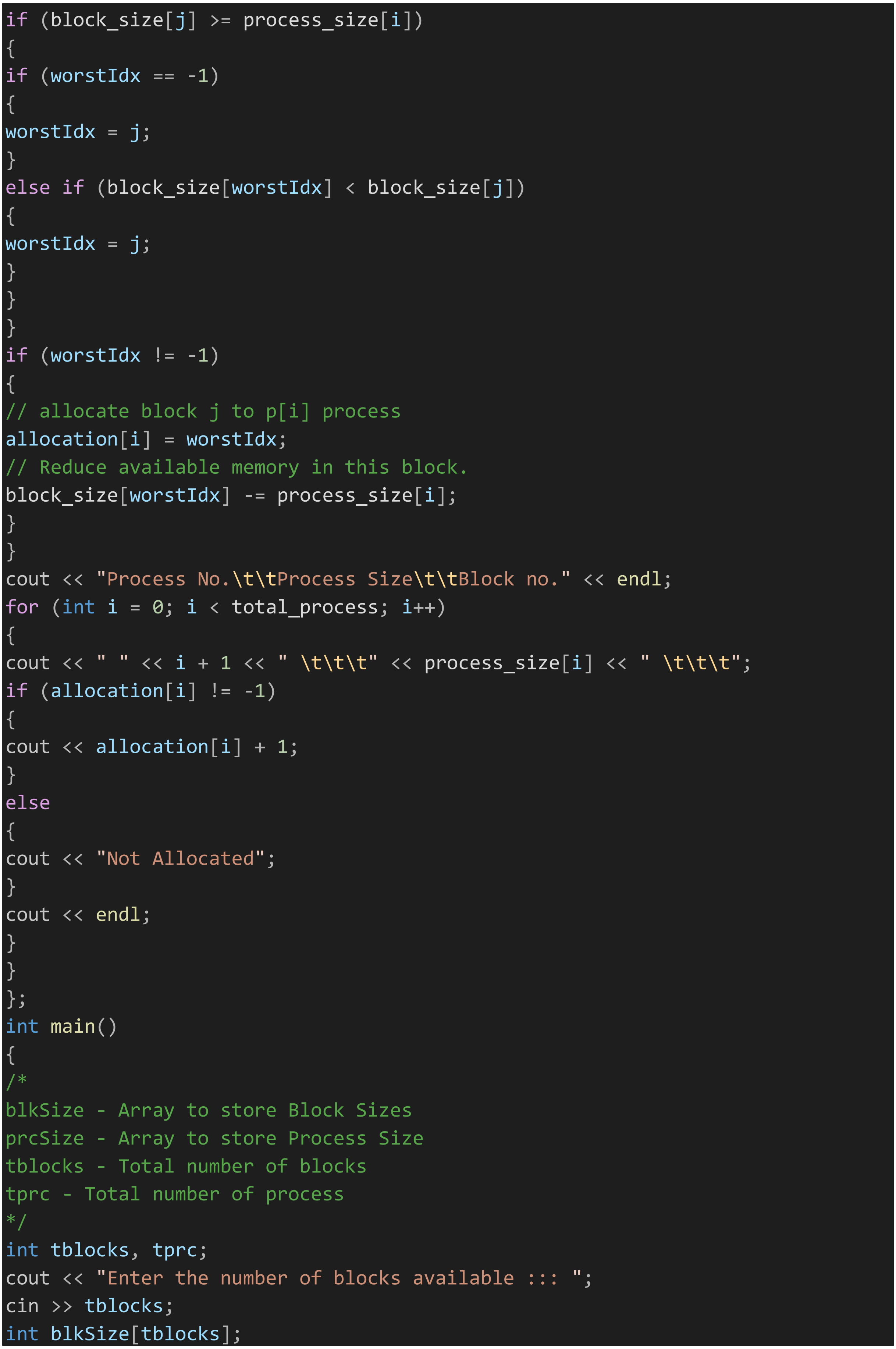
```
#include <iostream>
using namespace std;
class MemoryManagementAlgo
{
public:
int *block_size;
int total_blocks;
int *process_size;
int total_process;
MemoryManagementAlgo(int blkSize[], int tBlocks, int prSize[], int tProcess)
{
block_size = blkSize;
total_blocks = tBlocks;
process_size = prSize;
total_process = tProcess;
```



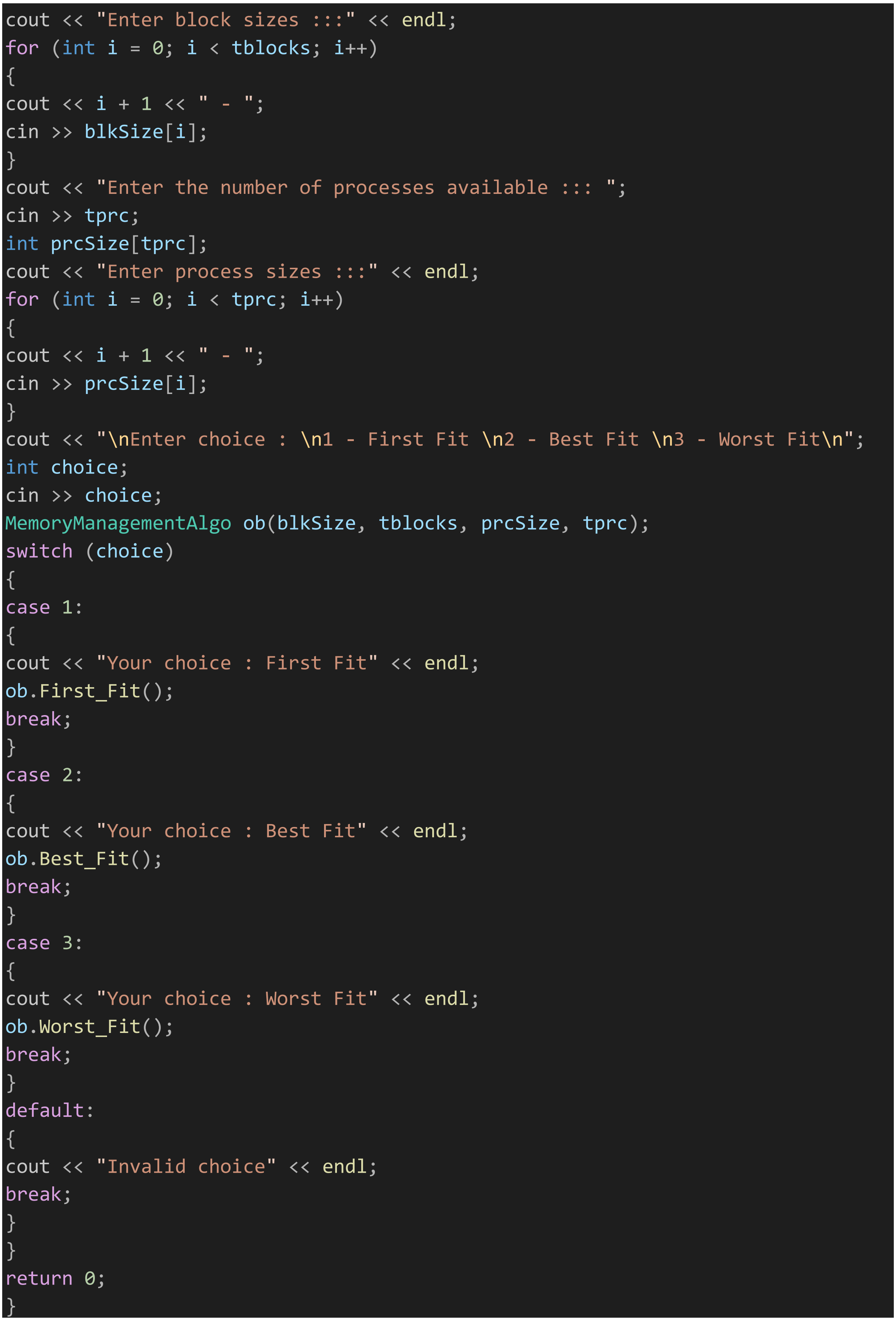
```
}
void First_Fit()
{
int allocation[total_process];
for (int i = 0; i < total_process; i++)
{
allocation[i] = -1;
}
for (int i = 0; i < total_process; i++)
{
for (int j = 0; j < total_blocks; j++)
{
if (block_size[j] >= process_size[i])
{
allocation[i] = j;
block_size[j] -= process_size[i];
break;
}
}
}
cout << "Process No.\t\tProcess Size\t\tBlock no." << endl;
for (int i = 0; i < total_process; i++)
{
cout << " " << i + 1 << " \t\t\t" << process_size[i] << " \t\t\t";
if (allocation[i] != -1)
{
cout << allocation[i] + 1;
}
else
{
cout << "Not Allocated";
}
cout << endl;
}
}
void Best_Fit()
{
int allocation[total_process];
for (int i = 0; i < total_process; i++)
{
allocation[i] = -1;
}
for (int i = 0; i < total_process; i++)
{
// Find the best fit block for current process
int bestIdx = -1;
for (int j = 0; j < total_blocks; j++)
{
```



```
if (block_size[j] >= process_size[i])
{
if (bestIdx == -1)
{
bestIdx = j;
}
else if (block_size[bestIdx] > block_size[j])
{
bestIdx = j;
}
}
}
if (bestIdx != -1)
{
// allocate block j to p[i] process
allocation[i] = bestIdx;
// Reduce available memory in this block.
block_size[bestIdx] -= process_size[i];
}
}
cout << "Process No.\t\tProcess Size\t\tBlock no." << endl;
for (int i = 0; i < total_process; i++)
{
cout << " " << i + 1 << " \t\t\t" << process_size[i] << " \t\t\t";
if (allocation[i] != -1)
{
cout << allocation[i] + 1;
}
else
{
cout << "Not Allocated";
}
cout << endl;
}
}
void Worst_Fit()
{
int allocation[total_process];
for (int i = 0; i < total_process; i++)
{
allocation[i] = -1;
}
for (int i = 0; i < total_process; i++)
{
// Find the best fit block for current process
int worstIdx = -1;
for (int j = 0; j < total_blocks; j++)
{
```

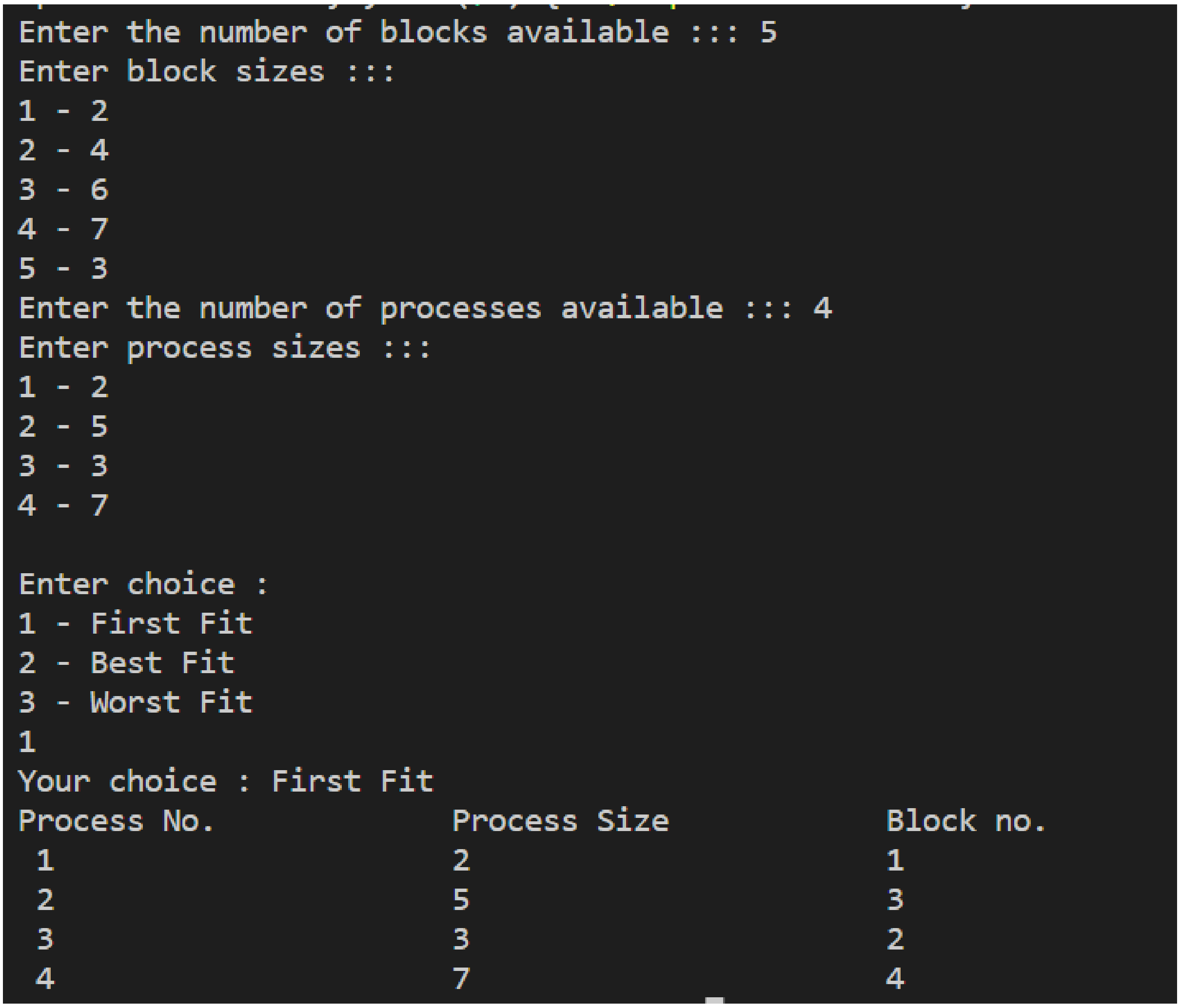


```
if (block_size[j] >= process_size[i])
{
if (worstIdx == -1)
{
worstIdx = j;
}
else if (block_size[worstIdx] < block_size[j])
{
worstIdx = j;
}
}
}
if (worstIdx != -1)
{
// allocate block j to p[i] process
allocation[i] = worstIdx;
// Reduce available memory in this block.
block_size[worstIdx] -= process_size[i];
}
}
cout << "Process No.\t\tProcess Size\t\tBlock no." << endl;
for (int i = 0; i < total_process; i++)
{
cout << " " << i + 1 << " \t\t\t" << process_size[i] << " \t\t\t";
if (allocation[i] != -1)
{
cout << allocation[i] + 1;
}
else
{
cout << "Not Allocated";
}
cout << endl;
}
}
};
int main()
{
/*
blkSize - Array to store Block Sizes
prcSize - Array to store Process Size
tblocks - Total number of blocks
tprc - Total number of process
*/
int tblocks, tprc;
cout << "Enter the number of blocks available ::: ";
cin >> tblocks;
int blkSize[tblocks];
```



```cpp
cout << "Enter block sizes :::" << endl;
for (int i = 0; i < tblocks; i++)
{
cout << i + 1 << " - ";
cin >> blkSize[i];
}
cout << "Enter the number of processes available ::: ";
cin >> tprc;
int prcSize[tprc];
cout << "Enter process sizes :::" << endl;
for (int i = 0; i < tprc; i++)
{
cout << i + 1 << " - ";
cin >> prcSize[i];
}
cout << "\nEnter choice : \n1 - First Fit \n2 - Best Fit \n3 - Worst Fit\n";
int choice;
cin >> choice;
MemoryManagementAlgo ob(blkSize, tblocks, prcSize, tprc);
switch (choice)
{
case 1:
{
cout << "Your choice : First Fit" << endl;
ob.First_Fit();
break;
}
case 2:
{
cout << "Your choice : Best Fit" << endl;
ob.Best_Fit();
break;
}
case 3:
{
cout << "Your choice : Worst Fit" << endl;
ob.Worst_Fit();
break;
}
default:
{
cout << "Invalid choice" << endl;
break;
}
}
return 0;
}
```

Output:



```
Enter the number of blocks available ::: 5
Enter block sizes :::
1 - 2
2 - 4
3 - 6
4 - 7
5 - 3
Enter the number of processes available ::: 4
Enter process sizes :::
1 - 2
2 - 5
3 - 3
4 - 7

Enter choice :
1 - First Fit
2 - Best Fit
3 - Worst Fit
1
Your choice : First Fit
Process No.             Process Size            Block no.
 1                      2                       1
 2                      5                       3
 3                      3                       2
 4                      7                       4
```