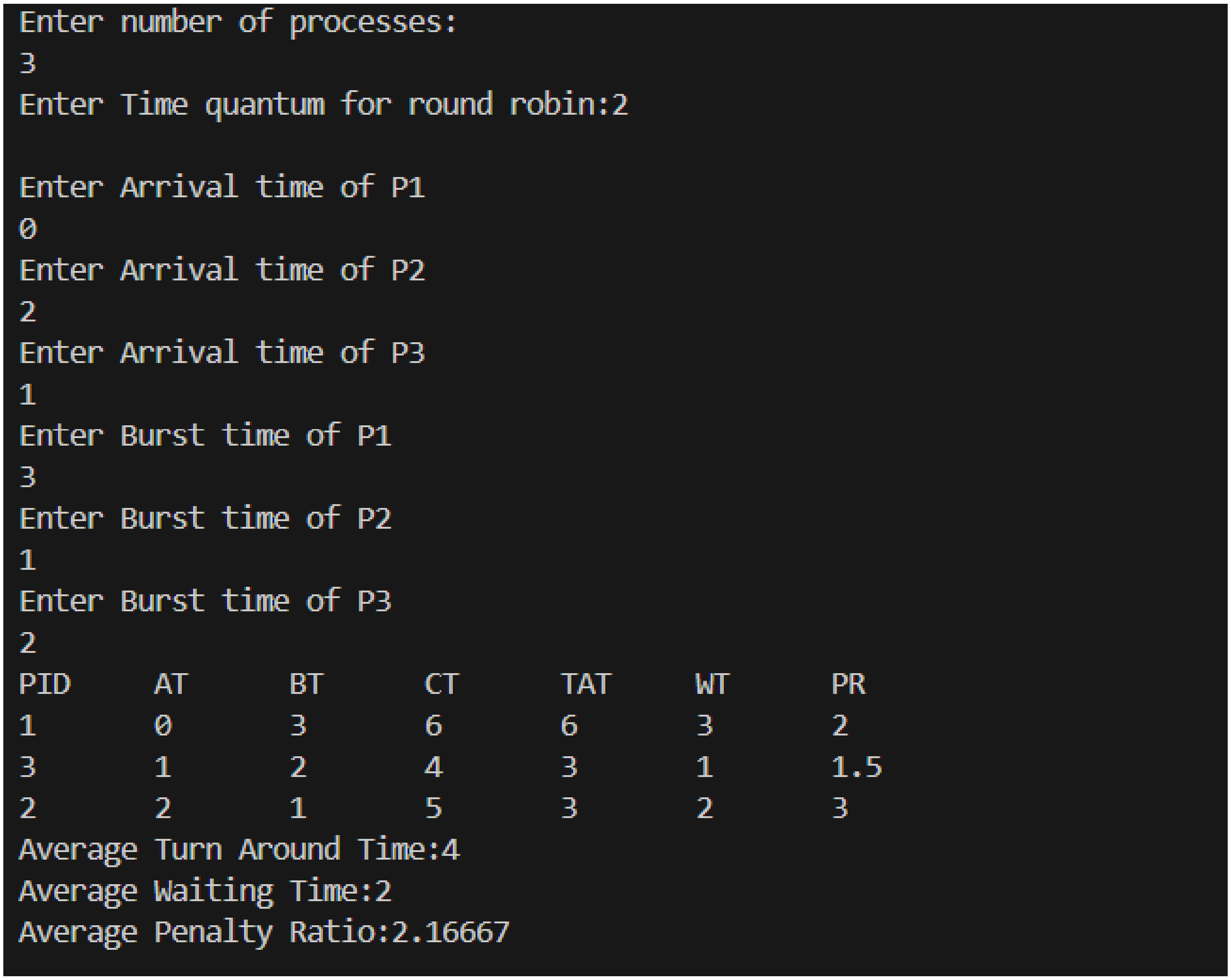
Sameer Gupta
OS Lab
October 31, 2025

Practical 6

Question 1. Write a CPP program to simulate the CPU scheduling algorithm Round Robin (RR)

```cpp
#include<bits/stdc++.h>
using namespace std;
void bubbleSort(vector<int>&at,vector<int>&bt,vector<int>&pid){
    int n=at.size();
    for(int i=0;i<n-1;i++){
        for(int j=0;j<n-1;j++){
            if(at[j]>at[j+1]){
                swap(at[j],at[j+1]);
                swap(pid[j],pid[j+1]);
                swap(bt[j],bt[j+1]);
            }
        }
    }
}
void readyQueue(queue<pair<int,int>>&rq,vector<bool>&remProcess,vector<int>&at,vector<int>&pid,int c
    for(int i=0;i<at.size();i++){
        if(at[i]<=currTime && !remProcess[i]){
            rq.push({pid[i],i});
            remProcess[i]=1;
        }
    }
}
int main(){
    int n;
    cout<<"Enter number of processes:"<<endl;
    cin>>n;
    vector<int>pid(n);
    vector<int>at(n);
    vector<int>bt(n);
    vector<int>ct(n);
    vector<int>tat(n);
    vector<int>wt(n);
    vector<float>pr(n);
    vector<bool>remProcess(n,0);
    queue<pair<int,int>>rq;
    int tq;
    cout<<"Enter Time quantum for round robin:";
    cin>>tq;
    cout<<endl;
    for(int i=0;i<n;i++){
        cout<<"Enter Arrival time of P"<<i+1<<endl;
        int num;
        cin>>num;
        at[i]=num;
        pid[i]=i+1;
    }
    for(int i=0;i<n;i++){
        cout<<"Enter Burst time of P"<<i+1<<endl;
        int num;
        cin>>num;
        bt[i]=num;
    }
    bubbleSort(at,bt,pid);
    vector<int>remTime(bt);
    int currTime=at[0];
    int completed=0;
    readyQueue(rq,remProcess,at,pid,currTime);
    while(completed<n){
        if(rq.empty()){
            int nextArrival=INT_MAX;
            for(int i=0;i<n;i++){
                if(!remProcess[i])nextArrival=min(nextArrival,at[i]);
            }
            if(nextArrival==INT_MAX)break;
            currTime=nextArrival;
            readyQueue(rq,remProcess,at,pid,currTime);
        }
        int getID=rq.front().first;
        int index=rq.front().second;
        rq.pop();
        if(remTime[index]>tq){
            remTime[index]-=tq;
            currTime+=tq;
            readyQueue(rq,remProcess,at,pid,currTime);
            rq.push({getID,index});
        }
        else{
            currTime+=remTime[index];
            readyQueue(rq,remProcess,at,pid,currTime);
            remTime[index]=0;
            ct[index]=currTime;
            completed++;
        }

    }
    for(int i=0;i<n;i++){
        tat[i]=ct[i]-at[i];
        wt[i]=tat[i]-bt[i];
        pr[i]=tat[i]*1.0/bt[i];
    }
    cout<<"PID\tAT\tBT\tCT\tTAT\tWT\tPR"<<endl;
    for(int i=0;i<n;i++){
        cout<<pid[i]<<"\t"<<at[i]<<"\t"<<bt[i]<<"\t"<<ct[i]<<"\t"<<tat[i]<<"\t"<<wt[i]<<"\t"<<pr[i]<
    }
    int sumtat=0,sumwt=0;
    float sumpr=0;
    for(int i=0;i<n;i++){
        sumtat+=tat[i];
        sumwt+=wt[i];
        sumpr+=pr[i];
    }
    cout<<"Average Turn Around Time:"<<1.0*sumtat/n<<endl;
    cout<<"Average Waiting Time:"<<1.0*sumwt/n<<endl;
    cout<<"Average Penalty Ratio:"<<sumpr/n<<endl;
}
```

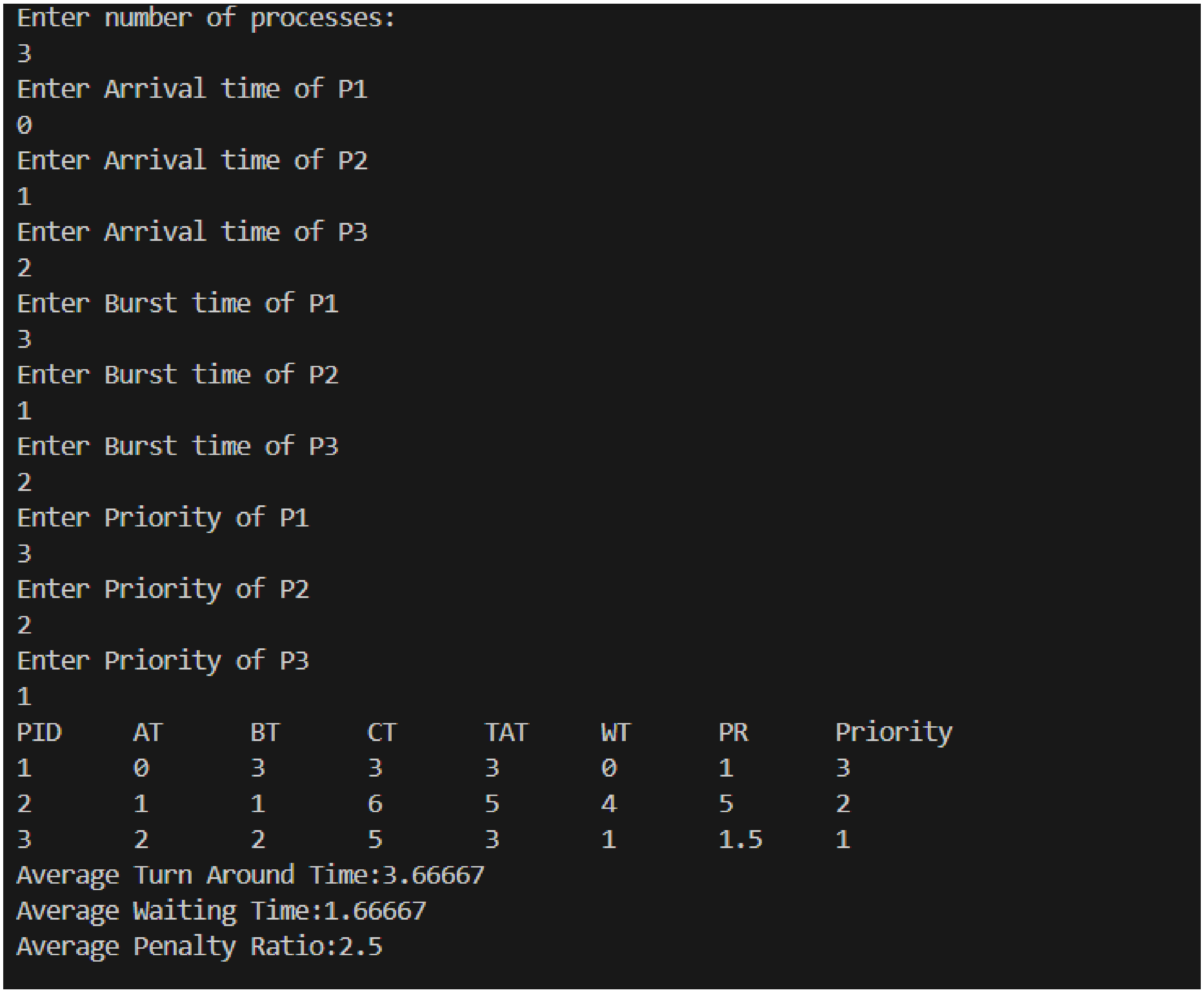


```
Enter number of processes:
3
Enter Time quantum for round robin:2

Enter Arrival time of P1
0
Enter Arrival time of P2
2
Enter Arrival time of P3
1
Enter Burst time of P1
3
Enter Burst time of P2
1
Enter Burst time of P3
2
PID     AT      BT      CT      TAT     WT      PR
1       0       3       6       6       3       2
3       1       2       4       3       1       1.5
2       2       1       5       3       2       3
Average Turn Around Time:4
Average Waiting Time:2
Average Penalty Ratio:2.16667
```

FIGURE 1. RR example output

Question 2. Write a CPP program to simulate the CPU scheduling algorithm Priority Scheduling.
a)Non Preemptive

```
#include<bits/stdc++.h>
using namespace std;
int getPID(vector<int>&pr,vector<int>&at,vector<int>&remProcess,int currTime){
    int n=remProcess.size();
    priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>>pq;
    for(int i=0;i<n;i++){
        int pid=remProcess[i];
        if(at[pid-1]<=currTime)pq.push({pr[pid-1],at[pid-1]*1000+pid});
    }
    if(pq.empty())return -1;
    int encoded=pq.top().second;
    return encoded%1000;
}

int main(){
    int n;
    cout<<"Enter number of processes:"<<endl;
    cin>>n;
    vector<int>pid(n);
    vector<int>at(n);
    vector<int>bt(n);
    vector<int>pr(n);
    vector<int>ct(n);
    vector<int>tat(n);
    vector<int>wt(n);
    vector<float>penalty(n);
    vector<int>remProcess(n);
    for(int i=0;i<n;i++){
        cout<<"Enter Arrival time of P"<<i+1<<endl;
        int num;
        cin>>num;
        at[i]=num;
        pid[i]=i+1;
        remProcess[i]=i+1;
    }
    for(int i=0;i<n;i++){
        cout<<"Enter Burst time of P"<<i+1<<endl;
        int num;
        cin>>num;
        bt[i]=num;
    }
    for(int i=0;i<n;i++){
        cout<<"Enter Priority of P"<<i+1<<endl;
        int num;
        cin>>num;
        pr[i]=num;
    }
    int currTime=*min_element(at.begin(),at.end());
    while(!remProcess.empty()){
        int getID=getPID(pr,at,remProcess,currTime);
        if(getID==-1){
            currTime++;
            continue;
        }
        for(int i=0;i<remProcess.size();i++){
            if(remProcess[i]==getID){
                remProcess.erase(remProcess.begin()+i);
                break;
            }
        }
        ct[getID-1]=currTime+bt[getID-1];
        currTime+=bt[getID-1];
    }
    for(int i=0;i<n;i++){
        tat[i]=ct[i]-at[i];
        wt[i]=tat[i]-bt[i];
        penalty[i]=tat[i]*1.0/bt[i];
    }
    cout<<"PID\tAT\tBT\tCT\tTAT\tWT\tPR\tPriority"<<endl;
    for(int i=0;i<n;i++){
        cout<<pid[i]<<"\t"<<at[i]<<"\t"<<bt[i]<<"\t"<<ct[i]<<"\t"<<tat[i]<<"\t"<<wt[i]<<"\t"<<penalt
    }
    int sumtat=0,sumwt=0;
    float sumpenalty=0;
    for(int i=0;i<n;i++){
        sumtat+=tat[i];
        sumwt+=wt[i];
        sumpenalty+=penalty[i];
    }
    cout<<"Average Turn Around Time:"<<1.0*sumtat/n<<endl;
    cout<<"Average Waiting Time:"<<1.0*sumwt/n<<endl;
    cout<<"Average Penalty Ratio:"<<sumpenalty/n<<endl;
}
```

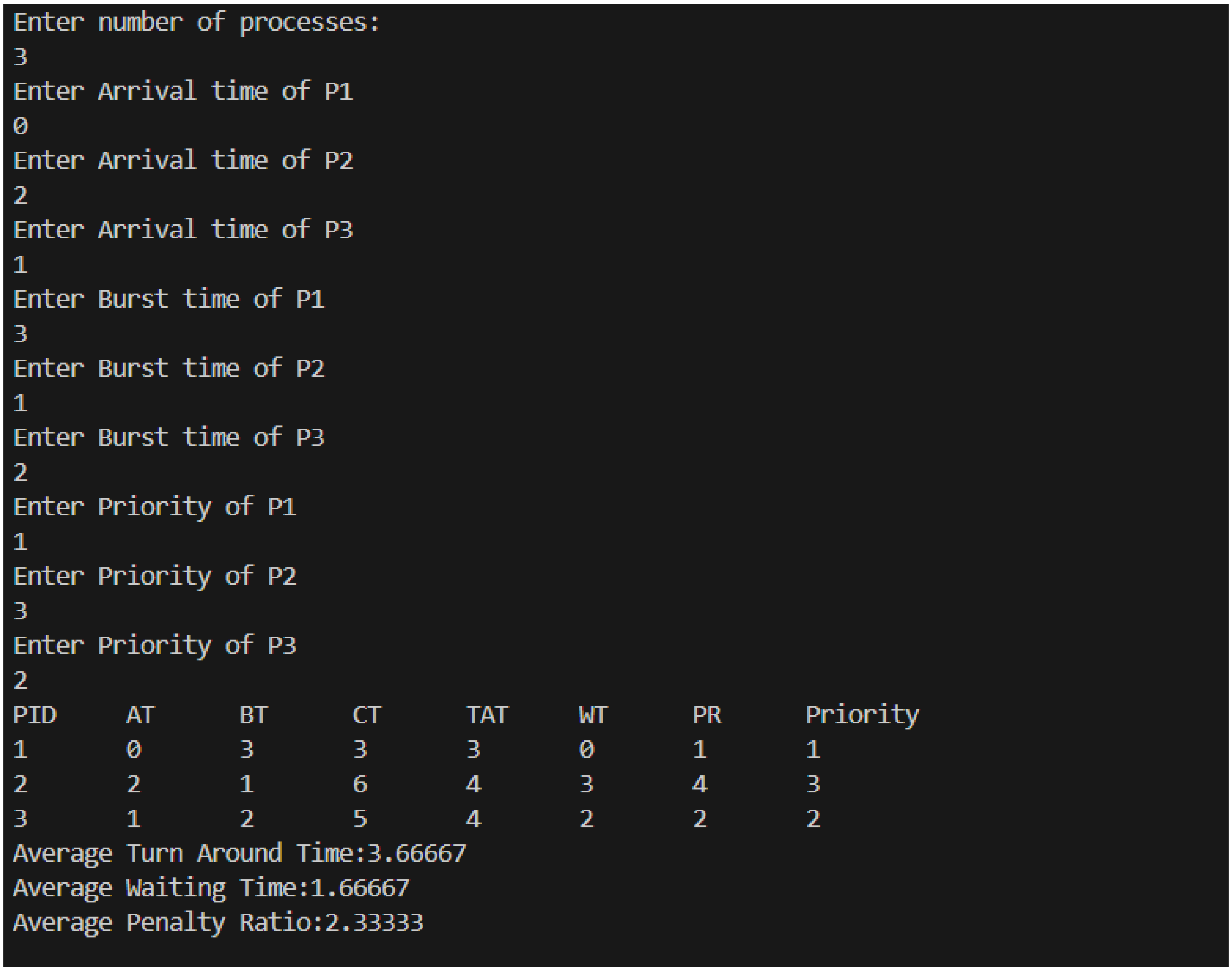


```
Enter number of processes:
3
Enter Arrival time of P1
0
Enter Arrival time of P2
1
Enter Arrival time of P3
2
Enter Burst time of P1
3
Enter Burst time of P2
1
Enter Burst time of P3
2
Enter Priority of P1
3
Enter Priority of P2
2
Enter Priority of P3
1
PID     AT      BT      CT      TAT     WT      PR      Priority
1       0       3       3       3       0       1       3
2       1       1       6       5       4       5       2
3       2       2       5       3       1       1.5     1
Average Turn Around Time:3.66667
Average Waiting Time:1.66667
Average Penalty Ratio:2.5
```

Figure 2. Priority Scheduling Non Preemptive example

b)Preemptive

```
#include<bits/stdc++.h>
using namespace std;
int getPID(vector<int>&pr,vector<int>&at,vector<int>&remProcess,int currTime){
    int n=remProcess.size();
    priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>>pq;
    for(int i=0;i<n;i++){
        int pid=remProcess[i];
        if(at[pid-1]<=currTime)pq.push({pr[pid-1],at[pid-1]*1000+pid});
    }
    if(pq.empty())return -1;
    int encoded=pq.top().second;
    return encoded%1000;
}

int main(){
    int n;
    cout<<"Enter number of processes:"<<endl;
    cin>>n;
    vector<int>pid(n);
    vector<int>at(n);
    vector<int>bt(n);
    vector<int>pr(n);
    vector<int>ct(n);
    vector<int>tat(n);
    vector<int>wt(n);
    vector<float>penalty(n);
    vector<int>remProcess(n);
    vector<int>remTime(n);
    for(int i=0;i<n;i++){
        cout<<"Enter Arrival time of P"<<i+1<<endl;
        int num;
        cin>>num;
        at[i]=num;
        pid[i]=i+1;
        remProcess[i]=i+1;
    }
    for(int i=0;i<n;i++){
        cout<<"Enter Burst time of P"<<i+1<<endl;
        int num;
        cin>>num;
        bt[i]=num;
        remTime[i]=num;
    }
    for(int i=0;i<n;i++){
        cout<<"Enter Priority of P"<<i+1<<endl;
        int num;
        cin>>num;
        pr[i]=num;
    }
    int currTime=*min_element(at.begin(),at.end());
    while(!remProcess.empty()){
        int getID=getPID(pr,at,remProcess,currTime);
        if(getID==-1){
            currTime++;
            continue;
        }
        remTime[getID-1]--;
        if(remTime[getID-1]==0){
            for(int i=0;i<remProcess.size();i++){
            if(remProcess[i]==getID){
                remProcess.erase(remProcess.begin()+i);
                break;
            }
            }
            ct[getID-1]=currTime+1;
        }
        currTime++;
    }
    for(int i=0;i<n;i++){
        tat[i]=ct[i]-at[i];
        wt[i]=tat[i]-bt[i];
        penalty[i]=tat[i]*1.0/bt[i];
    }
    cout<<"PID\tAT\tBT\tCT\tTAT\tWT\tPR\tPriority"<<endl;
    for(int i=0;i<n;i++){
        cout<<pid[i]<<"\t"<<at[i]<<"\t"<<bt[i]<<"\t"<<ct[i]<<"\t"<<tat[i]<<"\t"<<wt[i]<<"\t"<<penalt
    }
    int sumtat=0,sumwt=0;
    float sumpenalty=0;
    for(int i=0;i<n;i++){
        sumtat+=tat[i];
        sumwt+=wt[i];
        sumpenalty+=penalty[i];
    }
    cout<<"Average Turn Around Time:"<<1.0*sumtat/n<<endl;
    cout<<"Average Waiting Time:"<<1.0*sumwt/n<<endl;
    cout<<"Average Penalty Ratio:"<<sumpenalty/n<<endl;
}
```



```
Enter number of processes:
3
Enter Arrival time of P1
0
Enter Arrival time of P2
2
Enter Arrival time of P3
1
Enter Burst time of P1
3
Enter Burst time of P2
1
Enter Burst time of P3
2
Enter Priority of P1
1
Enter Priority of P2
3
Enter Priority of P3
2
PID     AT      BT      CT      TAT     WT      PR      Priority
1       0       3       3       3       0       1       1
2       2       1       6       4       3       4       3
3       1       2       5       4       2       2       2
Average Turn Around Time:3.66667
Average Waiting Time:1.66667
Average Penalty Ratio:2.33333
```

Figure 3. Priority Scheduling Preemptive example