CS266 ASSIGNMENT 6

NAME:

ARCHIT AGRAWAL

ROLL NO. :

202051213

SECTION:

2

Multilevel Feedback Queue Scheduling

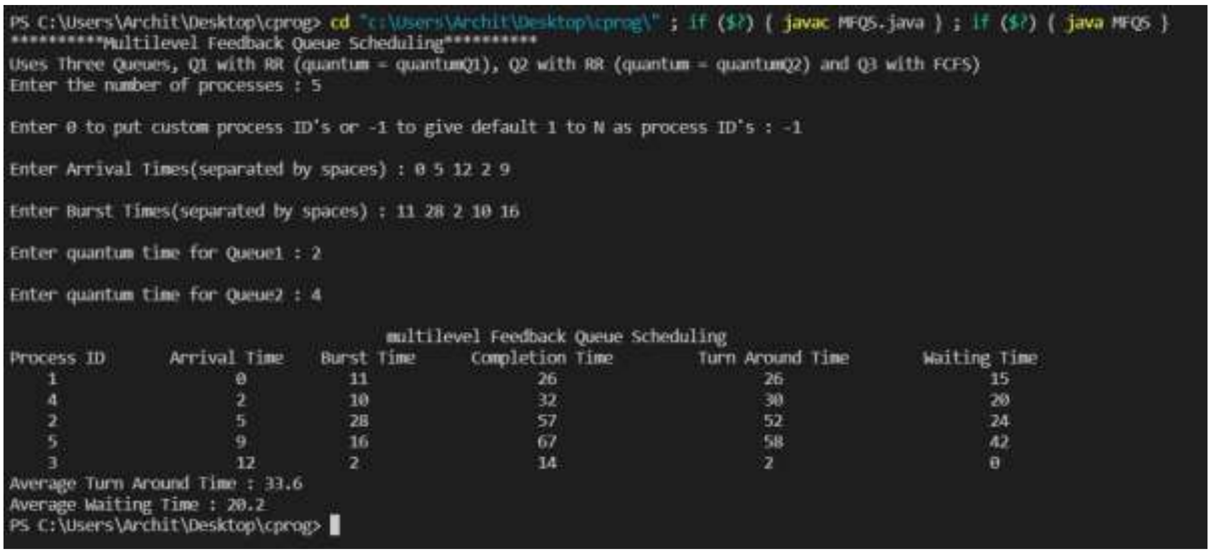
Question 1 and 2 have same scheduling algorithm but just with different quantum times for queues in the algorithm. Hence, there code is same, you just have to provide the quantum times during the input.

Code

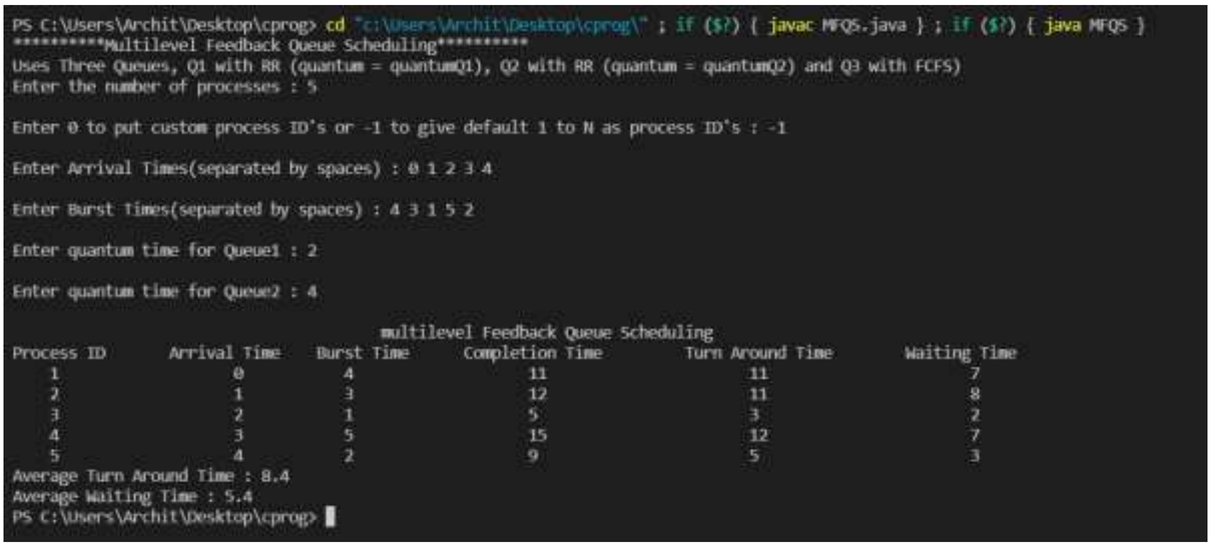
```java
/*
Name: Archit Agrawal
Student ID: 202051213

The following code is a simulation of the Multilevel Feedback Queue
Scheduling.
In this scheduling, Queue 1 has Round-Robin Scheduling(quantum = quantumQ1),
Queue has Round-Robin Scheduling(quantum = quantumQ2) and
Queue 3 has First Come First Serve.

The matrix mat stores all the data of the processes in the following code
    At index 0 -> Process ID
    At index 1 -> Arrival Time
    At index 2 -> Burst Time
    At index 3 -> Priority (the lower the number, the higher the priority)
    At index 4 -> finish or completion time
    At index 5 -> turn around time
    At index 6 -> waiting time

*/
import java.util.*;

public class MFQS{

    public static void arrangeArrival(int num, int[][] mat){
        for (int i = 0; i < num; i++) {
            for (int j = 0; j < num - i - 1; j++) {
                if (mat[j][1] > mat[j + 1][1]) {
                    for (int k = 0; k < 3; k++) {
                        int temp = mat[j][k];
                        mat[j][k] = mat[j + 1][k];
                        mat[j + 1][k] = temp;
                    }
                }
            }
        }
    }

    public static void completionTime(int num, int[][] mat, int quantumQ1, int
quantumQ2){
        Queue<Integer> q1 = new PriorityQueue<>();
        Queue<Integer> q2 = new PriorityQueue<>();
        Queue<Integer> q3 = new PriorityQueue<>();

        int[] rem = new int[num];
        for(int i = 0; i < num; i++) rem[i] = mat[i][2];

        int currTime = mat[0][1];
        int j = 0;
        int completed = 0;

        while(completed != num){

            while(j < num && mat[j][1] <= currTime){
                q1.add(j);
                mat[j][5] = 0;
                j++;
            }

            if(q1.peek() != null){

                if(rem[q1.peek()] <= quantumQ1){
                    currTime += rem[q1.peek()];
                    rem[q1.peek()] = -1;
                    completed++;
                    mat[q1.peek()][3] = currTime;
                    q1.poll();
                } else {
                    rem[q1.peek()] = rem[q1.peek()] - quantumQ1;
                    currTime += quantumQ1;
                    q2.add(q1.poll());
                }

            } else if(q1.peek() == null && q2.peek() != null){

                if(j < num && mat[j][1] - currTime >= quantumQ2){
                    if(rem[q2.peek()] <= quantumQ2){
                        currTime += rem[q2.peek()];
                        rem[q2.peek()] = -1;
                        completed++;
                        mat[q2.peek()][3] = currTime;
                        q2.poll();
                    } else {
                        rem[q2.peek()] = rem[q2.peek()] - quantumQ2;
                        currTime += quantumQ2;
                        q3.add(q2.poll());
                    }
                } else if(j < num && mat[j][1] - currTime < quantumQ2){
                    if(rem[q2.peek()] <= mat[j][1] - currTime){
                        currTime += rem[q2.peek()];
                        rem[q2.peek()] = -1;
                        completed++;
                        mat[q2.peek()][3] = currTime;
                        q2.poll();
                    } else {
                        rem[q2.peek()] = rem[q2.peek()] - (mat[j][1] -
currTime);
                        currTime += (mat[j][1] - currTime);
                        q3.add(q2.poll());
                    }
                } else{
                    if(rem[q2.peek()] <= quantumQ2){
                        currTime += rem[q2.peek()];
                        rem[q2.peek()] = -1;
                        completed++;
                        mat[q2.peek()][3] = currTime;
                        q2.poll();
                    } else {
                        rem[q2.peek()] = rem[q2.peek()] - quantumQ2;
                        currTime += quantumQ2;
                        q3.add(q2.poll());
                    }
                }
            } else if (q1.peek() == null && q2.peek() == null){
                if(q3.peek() != null){
                    if(j < num && mat[j][1] - currTime >= rem[q3.peek()]){
                        currTime += rem[q3.peek()];
                        rem[q3.peek()] = -1;
                        completed++;
                        mat[q3.peek()][3] = currTime;
                        q3.poll();
                    } else if(j < num && mat[j][1] - currTime <
rem[q3.peek()]){
                        rem[q3.peek()] = rem[q3.peek()] - (mat[j][1] -
currTime);
                        currTime += mat[j][1] - currTime;
                    } else{
                        currTime += rem[q3.peek()];
                        rem[q3.peek()] = -1;
                        completed++;
                        mat[q3.peek()][3] = currTime;
                        q3.poll();
                    }
                }
                if(j < num && j - completed < 1 && mat[j][1] > currTime){
                    currTime = mat[j][1];
                }
            }
        }

        turnAroundTime(num, mat);
        waitingTime(num, mat);
        printResult(num, mat);
    }

    public static void turnAroundTime(int num, int[][] mat){
        for(int i = 0; i < num; i++) mat[i][4] = mat[i][3] - mat[i][1];
    }

    public static void waitingTime(int num, int[][] mat){
        for(int i = 0; i < num; i++) mat[i][5] = mat[i][4] - mat[i][2];
    }

    public static double avgTAT(int num, int[][] mat){
        int sum = 0;
        for(int i = 0; i < num; i++) sum += mat[i][4];
        return (double)sum/num;
    }

    public static double avgWT(int num, int[][] mat){
        int sum = 0;
        for(int i = 0; i < num; i++) sum += mat[i][5];
        return (double)sum/num;
    }

    public static void MFQSScheduling(int num, int[][] mat, int quantumQ1, int
quantumQ2){
        arrangeArrival(num, mat);
        completionTime(num, mat, quantumQ1, quantumQ2);
    }

    public static void printResult(int num, int mat[][]){
        System.out.println("\t\t\t\t\tmultilevel Feedback Queue Scheduling");
        System.out.println("Process ID \t Arrival Time \t Burst Time \t
Completion Time \t Turn Around Time \t Waiting Time");
        for(int i = 0; i < num; i++){
            System.out.println("    " + mat[i][0] + "\t\t\t" + mat[i][1] +
"\t    " + mat[i][2] + "\t\t        " + mat[i][3] + "\t\t\t" + mat[i][4] +
"\t\t\t" + mat[i][5]);
        }
        System.out.println("Average Turn Around Time : " + avgTAT(num, mat));
        System.out.println("Average Waiting Time : " + avgWT(num, mat));
    }

    public static void main(String[] args){
        System.out.println("**********Multilevel Feedback Queue
Scheduling**********");
        System.out.println("Uses Three Queues, Q1 with RR (quantum =
quantumQ1), Q2 with RR (quantum = quantumQ2) and Q3 with FCFS)");
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of processes : ");
        int n = sc.nextInt();
        System.out.println();
        int[][] mat = new int[n][6];

        System.out.print("Enter 0 to put custom process ID's or -1 to give
default 1 to N as process ID's : ");
        int id = sc.nextInt();
        System.out.println();

        if(id == -1){
            for(int i = 0; i < n; i++) mat[i][0] = i + 1;
        } else {
            System.out.println("Enter Process ID's(separated by spaces) : ");
            for(int i = 0; i < n; i++) mat[i][0] = sc.nextInt();
        }

        System.out.print("Enter Arrival Times(separated by spaces) : ");
        for(int i = 0; i < n; i++) mat[i][1] = sc.nextInt();

        System.out.println();

        System.out.print("Enter Burst Times(separated by spaces) : ");
        for(int i = 0; i < n; i++) mat[i][2] = sc.nextInt();

        System.out.println();

        System.out.print("Enter quantum time for Queue1 : ");
        int quantQ1 = sc.nextInt();
        System.out.println();

        System.out.print("Enter quantum time for Queue2 : ");
        int quantQ2 = sc.nextInt();
        System.out.println();

        MFQSScheduling(n, mat, quantQ1, quantQ2);

    }
}
```

Output (Question 1)

In question 1, Q1 has RR(quantum 2) and Q2 has RR(quantum 4) and Q3 has FCFS.



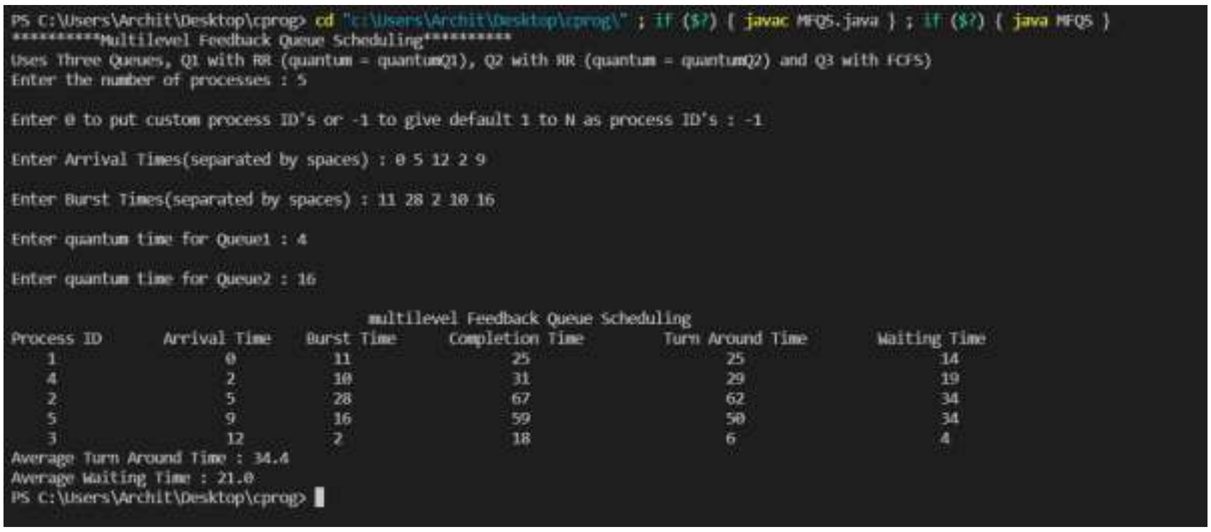
```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac MFQS.java } ; if ($?) { java MFQS }
**********Multilevel Feedback Queue Scheduling**********
Uses Three Queues, Q1 with RR (quantum = quantumQ1), Q2 with RR (quantum = quantumQ2) and Q3 with FCFS)
Enter the number of processes : 5

Enter 0 to put custom process ID's or -1 to give default 1 to N as process ID's : -1

Enter Arrival Times(separated by spaces) : 0 5 12 2 9

Enter Burst Times(separated by spaces) : 11 28 2 10 16

Enter quantum time for Queue1 : 2

Enter quantum time for Queue2 : 4

                                  multilevel Feedback Queue Scheduling
Process ID        Arrival Time    Burst Time     Completion Time        Turn Around Time        Waiting Time
    1                 0              11               26                      26                     15
    4                 2              10               32                      30                     20
    2                 5              28               57                      52                     24
    5                 9              16               67                      58                     42
    3                 12             2                14                      2                      0
Average Turn Around Time : 33.6
Average Waiting Time : 20.2
PS C:\Users\Archit\Desktop\cprog> 
```



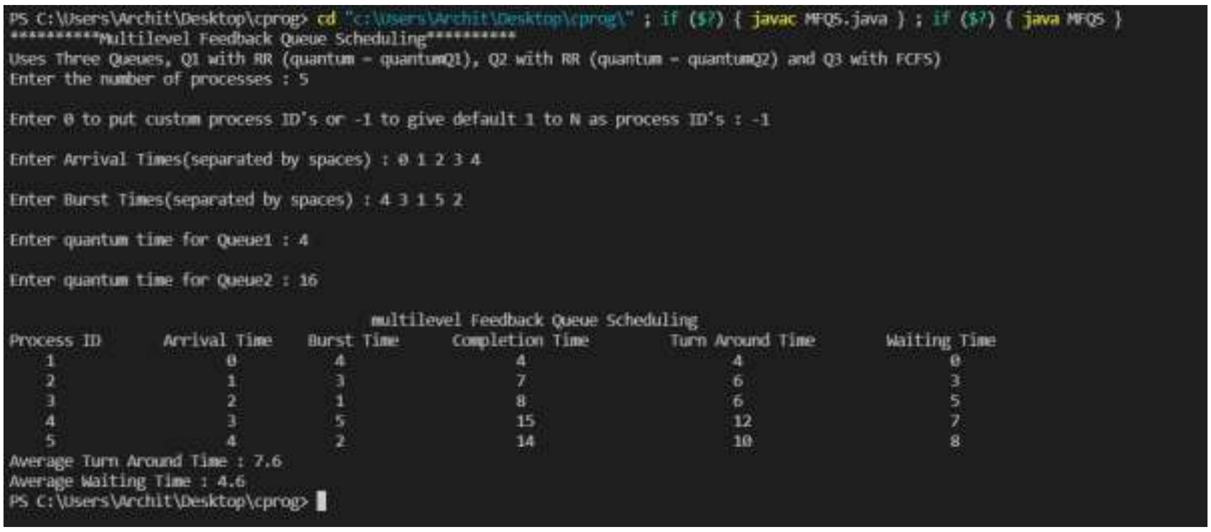
```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac MFQS.java } ; if ($?) { java MFQS }
**********Multilevel Feedback Queue Scheduling**********
Uses Three Queues, Q1 with RR (quantum = quantumQ1), Q2 with RR (quantum = quantumQ2) and Q3 with FCFS)
Enter the number of processes : 5

Enter 0 to put custom process ID's or -1 to give default 1 to N as process ID's : -1

Enter Arrival Times(separated by spaces) : 0 1 2 3 4

Enter Burst Times(separated by spaces) : 4 3 1 5 2

Enter quantum time for Queue1 : 2

Enter quantum time for Queue2 : 4

                                  multilevel Feedback Queue Scheduling
Process ID        Arrival Time    Burst Time     Completion Time        Turn Around Time        Waiting Time
    1                 0              4                11                      11                     7
    2                 1              3                12                      11                     8
    3                 2              1                5                       3                      2
    4                 3              5                15                      12                     7
    5                 4              2                9                       5                      3
Average Turn Around Time : 8.4
Average Waiting Time : 5.4
PS C:\Users\Archit\Desktop\cprog> 
```

Output (Question 2)

In question 2, Q1 has RR(quantum 4) and Q2 has RR(quantum 16) and Q3 has FCFS.



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac MFQS.java } ; if ($?) { java MFQS }
**********Multilevel Feedback Queue Scheduling**********
Uses Three Queues, Q1 with RR (quantum = quantumQ1), Q2 with RR (quantum = quantumQ2) and Q3 with FCFS)
Enter the number of processes : 5

Enter 0 to put custom process ID's or -1 to give default 1 to N as process ID's : -1

Enter Arrival Times(separated by spaces) : 0 5 12 2 9

Enter Burst Times(separated by spaces) : 11 28 2 10 16

Enter quantum time for Queue1 : 4

Enter quantum time for Queue2 : 16

                        multilevel Feedback Queue Scheduling
Process ID      Arrival Time    Burst Time      Completion Time         Turn Around Time        Waiting Time
    1                0             11                25                       25                     14
    4                2             10                31                       29                     19
    2                5             28                67                       62                     34
    5                9             16                59                       50                     34
    3               12             2                 18                       6                      4
Average Turn Around Time : 34.4
Average Waiting Time : 21.0
PS C:\Users\Archit\Desktop\cprog> █
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac MFQS.java } ; if ($?) { java MFQS }
**********Multilevel Feedback Queue Scheduling**********
Uses Three Queues, Q1 with RR (quantum = quantumQ1), Q2 with RR (quantum = quantumQ2) and Q3 with FCFS)
Enter the number of processes : 5

Enter 0 to put custom process ID's or -1 to give default 1 to N as process ID's : -1

Enter Arrival Times(separated by spaces) : 0 1 2 3 4

Enter Burst Times(separated by spaces) : 4 3 1 5 2

Enter quantum time for Queue1 : 4

Enter quantum time for Queue2 : 16

                        multilevel Feedback Queue Scheduling
Process ID      Arrival Time    Burst Time      Completion Time         Turn Around Time        Waiting Time
    1                0             4                 4                        4                      0
    2                1             3                 7                        6                      3
    3                2             1                 8                        6                      5
    4                3             5                 15                       12                     7
    5                4             2                 14                       10                     8
Average Turn Around Time : 7.6
Average Waiting Time : 4.6
PS C:\Users\Archit\Desktop\cprog> █
```

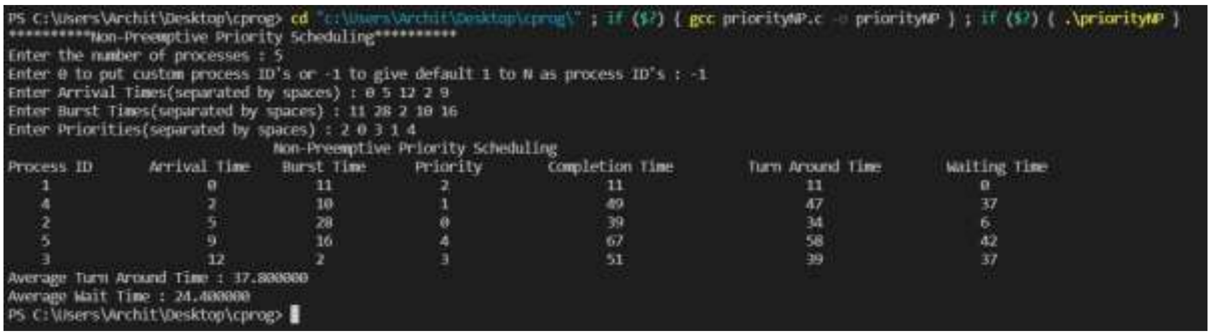
Priority Scheduling (Non-Preemptive)

Code

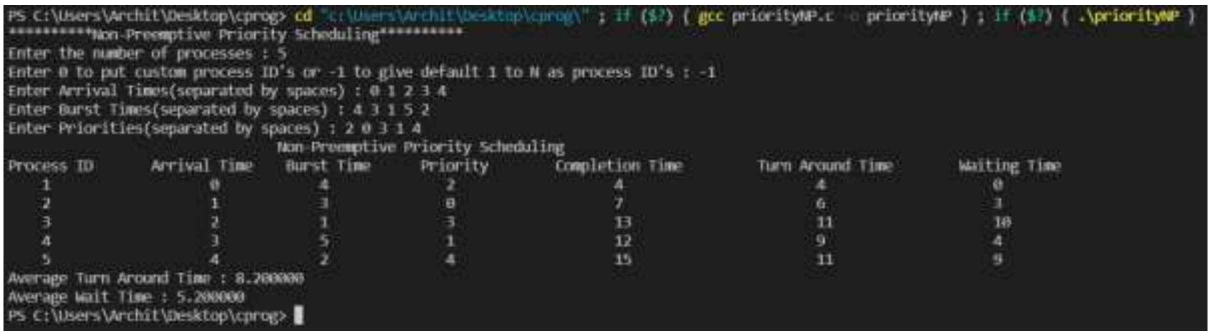
```
/*
Name: Archit Agrawal
Student ID: 202051213

The following code is a simulation of the Non-Preemptive Priority Scheduling
Documentation:
The matrix mat stores all the data of the processes in the following code
    At index 0 -> Process ID
    At index 1 -> Arrival Time
    At index 2 -> Burst Time
    At index 3 -> Priority (the lower the number, the higher the priority)
    At index 4 -> finish or completion time
    At index 5 -> turn around time
    At index 6 -> waiting time

*/


#include<stdio.h>

//function to arrange processes by their arrival times
void arrangeArrival(int num, int mat[][7]);
//function to calculate completion time for each process
void completionTime(int num, int mat[][7]);
//function to calculate TAT for each process
void turnAroundTime(int num, int mat[][7]);
//function to calculate WT for each process
void waitingTime(int num, int mat[][7]);
//function to calculate average TAT
double avgTAT(int num, int mat[][7]);
//function to calculate average WT
double avgWT(int num, int mat[][7]);
//scheduling function that the user need to call
void priorityScheduling(int num, int mat[][7]);
//utility function to print the result in formatted way
void printResult(int num, int mat[][7]);

void arrangeArrival(int num, int mat[][7]){
    for (int i = 0; i < num; i++) {
        for (int j = 0; j < num - i - 1; j++) {
            if (mat[j][1] > mat[j + 1][1]) {
```

```c
                for (int k = 0; k < 4; k++) {

                    int temp = mat[j][k];
                    mat[j][k] = mat[j + 1][k];
                    mat[j + 1][k] = temp;
                }
            }
        }
    }

}

void completionTime(int num, int mat[][7]){
    int arr[num];
    for(int i = 0; i < num; i++) arr[i] = 0;
    int j = 0;
    int currTime = mat[0][1];
    int completed = 0;

    while(completed != num){
        while(j < num && arr[j] != -1 && mat[j][1] <= currTime){
            arr[j] = 1;
            j++;
        }

        //for(int s = 0; s <num; s++) printf("%d, ", arr[s]);
        //printf("currTime : %d, j = %d, completed = %d\n", currTime, j,
completed);

        int maxPriorAmongArrived = 0;
        for(int k = 0; k < num; k++){
            if(arr[k] == 1){
                maxPriorAmongArrived = k;
                break;
            }
        }

        for(int k = maxPriorAmongArrived + 1; k < num; k++){
            if(arr[k] == 1 && mat[k][3] < mat[maxPriorAmongArrived][3]){
                maxPriorAmongArrived = k;
            }
        }

        currTime += mat[maxPriorAmongArrived][2];
        mat[maxPriorAmongArrived][4] = currTime;
        completed++;
        arr[maxPriorAmongArrived] = -1;
        //for(int s = 0; s <num; s++) printf("%d, ", arr[s]);
        //printf("currTime : %d, j = %d, completed = %d\n", currTime, j,
completed);
        if(j < num && j - completed < 1 && mat[j][1] > currTime){
            currTime = mat[j][1];
        }


    }
    //printf("Completion Time Done");
    turnAroundTime(num, mat);
    //printf("TAT Done");
    waitingTime(num, mat);
    //printf("WT Done");
    printResult(num, mat);
}

void turnAroundTime(int num, int mat[][7]){
    for(int i = 0; i < num; i++) mat[i][5] = mat[i][4] - mat[i][1];
}

void waitingTime(int num, int mat[][7]){
    for(int i = 0; i < num; i++) mat[i][6] = mat[i][5] - mat[i][2];
}

double avgTAT(int num, int mat[][7]){
    int sum = 0;
    for(int i = 0; i < num; i++) sum += mat[i][5];
    return (double)sum/num;
}

double avgWT(int num, int mat[][7]){
    int sum = 0;
    for(int i = 0; i < num; i++) sum += mat[i][6];
    return (double)sum/num;
}

void priorityScheduling(int num, int mat[][7]){
    arrangeArrival(num, mat);
    //arrangePriority(num, mat);
    completionTime(num, mat);
}

void printResult(int num, int mat[][7]){
    printf("\t\t\t\tNon-Preemptive Priority Scheduling\n");
    printf("Process ID \t Arrival Time \t Burst Time \t Priority \t Completion
Time \t Turn Around Time \t Waiting Time\n");
    for(int i = 0; i < num; i++){
        printf("    %d\t\t\t%d\t     %d\t\t    %d    \t\t%d\t\t\t%d\t\t     %d
\n", mat[i][0], mat[i][1], mat[i][2], mat[i][3], mat[i][4], mat[i][5],
mat[i][6]);
    }

    printf("Average Turn Around Time : %f\n", avgTAT(num, mat));
    printf("Average Wait Time : %f\n", avgWT(num, mat));
}

int main(){
    printf("**********Non-Preemptive Priority Scheduling**********\n");
    int n;
    printf("Enter the number of processes : ");
    scanf("%d", &n);
    int mat[n][7];

    printf("Enter 0 to put custom process ID's or -1 to give default 1 to N as
process ID's : ");
    int id;
    scanf("%d", &id);

    if(id == -1){
        for(int i = 0; i < n; i++) mat[i][0] = i + 1;
    } else {
        printf("Enter Process ID's(separated by spaces) : ");
        for(int i = 0; i < n; i++){
            int s;
            scanf("%d", &s);
            mat[i][0] = s;
        }
    }

    printf("Enter Arrival Times(separated by spaces) : ");

    for(int i = 0; i < n; i++){
        int s;
        scanf("%d", &s);
        mat[i][1] = s;
    }

    printf("Enter Burst Times(separated by spaces) : ");

    for(int i = 0; i < n; i++){
        int s;
        scanf("%d", &s);
        mat[i][2] = s;
    }

    printf("Enter Priorities(separated by spaces) : ");

    for(int i = 0; i < n; i++){
        int s;
        scanf("%d", &s);
        mat[i][3] = s;
    }

    priorityScheduling(n, mat);
}
```

<u>Output</u>



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { gcc priorityNP.c -o priorityNP } ; if ($?) { .\priorityNP }
**********Non-Preemptive Priority Scheduling**********
Enter the number of processes : 5
Enter 0 to put custom process ID's or -1 to give default 1 to N as process ID's : -1
Enter Arrival Times(separated by spaces) : 0 5 12 2 9
Enter Burst Times(separated by spaces) : 11 28 2 10 16
Enter Priorities(separated by spaces) : 2 0 3 1 4
                                Non-Preemptive Priority Scheduling
Process ID       Arrival Time    Burst Time      Priority        Completion Time         Turn Around Time        Waiting Time
    1                   0           11              2                  11                      11                     0
    4                   2           10              1                  49                      47                     37
    2                   5           28              0                  39                      34                     6
    5                   9           16              4                  67                      58                     42
    3                   12          2               3                  51                      39                     37
Average Turn Around Time : 37.800000
Average Wait Time : 24.400000
PS C:\Users\Archit\Desktop\cprog> 
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { gcc priorityNP.c -o priorityNP } ; if ($?) { .\priorityNP }
**********Non-Preemptive Priority Scheduling**********
Enter the number of processes : 5
Enter 0 to put custom process ID's or -1 to give default 1 to N as process ID's : -1
Enter Arrival Times(separated by spaces) : 0 1 2 3 4
Enter Burst Times(separated by spaces) : 4 3 1 5 2
Enter Priorities(separated by spaces) : 2 0 3 1 4
                                Non-Preemptive Priority Scheduling
Process ID       Arrival Time    Burst Time      Priority        Completion Time         Turn Around Time        Waiting Time
    1                   0           4               2                  4                       4                      0
    2                   1           3               0                  7                       6                      3
    3                   2           1               3                  13                      11                     10
    4                   3           5               1                  12                      9                      4
    5                   4           2               4                  15                      11                     9
Average Turn Around Time : 8.200000
Average Wait Time : 5.200000
PS C:\Users\Archit\Desktop\cprog> 
```

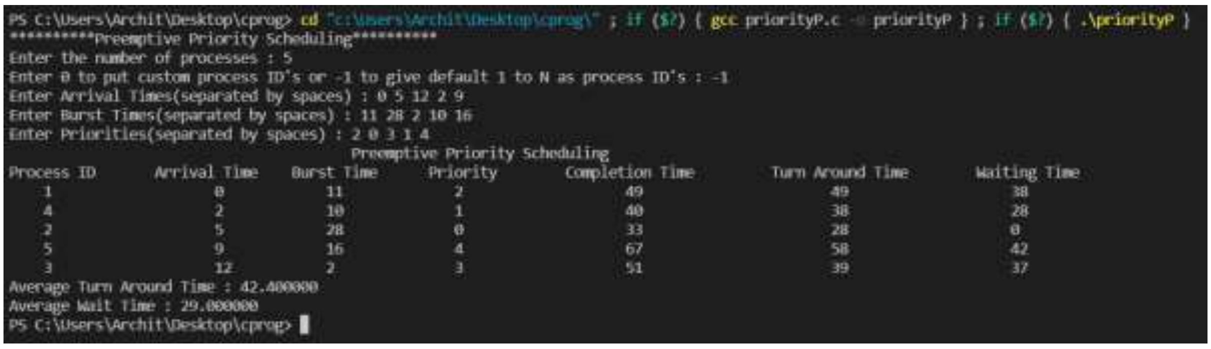
Priority Scheduling (Preemptive)

Code

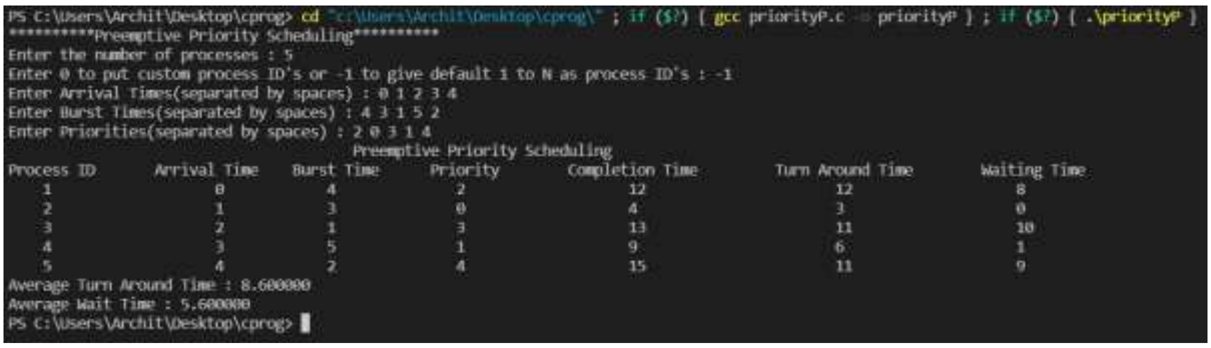
```
/*
Name: Archit Agrawal
Student ID: 202051213

The following code is a simulation of the Preemptive Priority Scheduling
Documentation:
The matrix mat stores all the data of the processes in the following code
    At index 0 -> Process ID
    At index 1 -> Arrival Time
    At index 2 -> Burst Time
    At index 3 -> Priority (the lower the number, the higher the priority)
    At index 4 -> finish or completion time
    At index 5 -> turn around time
    At index 6 -> waiting time

*/


#include<stdio.h>

//function to arrange processes by their arrival times
void arrangeArrival(int num, int mat[][7]);
//function to calculate completion time for each process
void completionTime(int num, int mat[][7]);
//function to calculate TAT for each process
void turnAroundTime(int num, int mat[][7]);
//function to calculate WT for each process
void waitingTime(int num, int mat[][7]);
//function to calculate average TAT
double avgTAT(int num, int mat[][7]);
//function to calculate average WT
double avgWT(int num, int mat[][7]);
//scheduling function that the user need to call
void priorityScheduling(int num, int mat[][7]);
//utility function to print the result in formatted way
void printResult(int num, int mat[][7]);

void arrangeArrival(int num, int mat[][7]){
    for (int i = 0; i < num; i++) {
        for (int j = 0; j < num - i - 1; j++) {
            if (mat[j][1] > mat[j + 1][1]) {
                for (int k = 0; k < 4; k++) {

                    int temp = mat[j][k];
                    mat[j][k] = mat[j + 1][k];
                    mat[j + 1][k] = temp;
                }
            }
        }
    }
}

void completionTime(int num, int mat[][7]){
    int arr[num];
    int rem[num];

    for(int i = 0; i < num; i++) arr[i] = 0;
    for(int i = 0; i < num; i++) rem[i] = mat[i][2];
    int j = 0;
    int currTime = mat[0][1];
    int completed = 0;

    while(completed != num){

        while(j < num && arr[j] != -1 && mat[j][1] <= currTime){
            arr[j] = 1;
            j++;
        }

        //for(int s = 0; s <num; s++) printf("%d, ", arr[s]);
        //printf("currTime : %d, j = %d, completed = %d\n", currTime, j,
completed);

        int maxPriorAmongArrived = 0;
        for(int k = 0; k < num; k++){
            if(arr[k] == 1){
                maxPriorAmongArrived = k;
                break;
            }
        }

        for(int k = maxPriorAmongArrived + 1; k < num; k++){
            if(arr[k] == 1 && mat[k][3] < mat[maxPriorAmongArrived][3]){
                maxPriorAmongArrived = k;
            }
        }

        if(j <  num && mat[j][1] - currTime >= rem[maxPriorAmongArrived]){
            currTime += rem[maxPriorAmongArrived];
            rem[maxPriorAmongArrived] = -1;
            completed++;
            mat[maxPriorAmongArrived][4] = currTime;
            arr[maxPriorAmongArrived] = -1;
        } else if(j < num && mat[j][1] - currTime <
rem[maxPriorAmongArrived]){
            rem[maxPriorAmongArrived] = rem[maxPriorAmongArrived] - (mat[j][1]
- currTime);
            currTime = mat[j][1];
        } else{
            currTime += rem[maxPriorAmongArrived];
            rem[maxPriorAmongArrived] = -1;
            completed++;
            mat[maxPriorAmongArrived][4] = currTime;
            arr[maxPriorAmongArrived] = -1;
        }

        if(j < num && j - completed < 1 && mat[j][1] > currTime){
            currTime = mat[j][1];
        }
    }
    //printf("Completion Time Done");
    turnAroundTime(num, mat);
    //printf("TAT Done");
    waitingTime(num, mat);
    //printf("WT Done");
    printResult(num, mat);
}

void turnAroundTime(int num, int mat[][7]){
    for(int i = 0; i < num; i++) mat[i][5] = mat[i][4] - mat[i][1];
}

void waitingTime(int num, int mat[][7]){
    for(int i = 0; i < num; i++) mat[i][6] = mat[i][5] - mat[i][2];
}

double avgTAT(int num, int mat[][7]){
    int sum = 0;
    for(int i = 0; i < num; i++) sum += mat[i][5];
    return (double)sum/num;
}

double avgWT(int num, int mat[][7]){
    int sum = 0;
    for(int i = 0; i < num; i++) sum += mat[i][6];
```

```c
    return (double)sum/num;
}

void priorityScheduling(int num, int mat[][7]){
    arrangeArrival(num, mat);
    completionTime(num, mat);
}

void printResult(int num, int mat[][7]){
    printf("\t\t\t\t\tPreemptive Priority Scheduling\n");
    printf("Process ID \t Arrival Time \t Burst Time \t Priority \t Completion
Time \t Turn Around Time \t Waiting Time\n");
    for(int i = 0; i < num; i++){
        printf("    %d\t\t\t%d\t     %d\t\t    %d    \t\t%d\t\t\t%d\t\t     %d
\n", mat[i][0], mat[i][1], mat[i][2], mat[i][3], mat[i][4], mat[i][5],
mat[i][6]);
    }

    printf("Average Turn Around Time : %f\n", avgTAT(num, mat));
    printf("Average Wait Time : %f\n", avgWT(num, mat));
}

int main(){
    printf("**********Preemptive Priority Scheduling**********\n");
    int n;
    printf("Enter the number of processes : ");
    scanf("%d", &n);
    int mat[n][7];

    printf("Enter 0 to put custom process ID's or -1 to give default 1 to N as
process ID's : ");
    int id;
    scanf("%d", &id);

    if(id == -1){
        for(int i = 0; i < n; i++) mat[i][0] = i + 1;
    } else {
        printf("Enter Process ID's(separated by spaces) : ");
        for(int i = 0; i < n; i++){
            int s;
            scanf("%d", &s);
            mat[i][0] = s;
        }
    }

    printf("Enter Arrival Times(separated by spaces) : ");
    for(int i = 0; i < n; i++){
        int s;
        scanf("%d", &s);
        mat[i][1] = s;
    }

    printf("Enter Burst Times(separated by spaces) : ");

    for(int i = 0; i < n; i++){
        int s;
        scanf("%d", &s);
        mat[i][2] = s;
    }

    printf("Enter Priorities(separated by spaces) : ");

    for(int i = 0; i < n; i++){
        int s;
        scanf("%d", &s);
        mat[i][3] = s;
    }

    priorityScheduling(n, mat);
}
```

Output



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { gcc priorityP.c -o priorityP } ; if ($?) { .\priorityP }
**********Preemptive Priority Scheduling**********
Enter the number of processes : 5
Enter 0 to put custom process ID's or -1 to give default 1 to N as process ID's : -1
Enter Arrival Times(separated by spaces) : 0 5 12 2 9
Enter Burst Times(separated by spaces) : 11 28 2 10 16
Enter Priorities(separated by spaces) : 2 0 3 1 4
                                          Preemptive Priority Scheduling
Process ID        Arrival Time    Burst Time      Priority        Completion Time          Turn Around Time         Waiting Time
    1                   0             11              2                 49                       49                      38
    4                   2             10              1                 40                       38                      28
    2                   5             28              0                 33                       28                      0
    5                   9             16              4                 67                       58                      42
    3                   12            2               3                 51                       39                      37
Average Turn Around Time : 42.400000
Average Wait Time : 29.000000
PS C:\Users\Archit\Desktop\cprog> 
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { gcc priorityP.c -o priorityP } ; if ($?) { .\priorityP }
**********Preemptive Priority Scheduling**********
Enter the number of processes : 5
Enter 0 to put custom process ID's or -1 to give default 1 to N as process ID's : -1
Enter Arrival Times(separated by spaces) : 0 1 2 3 4
Enter Burst Times(separated by spaces) : 4 3 1 5 2
Enter Priorities(separated by spaces) : 2 0 3 1 4
                                          Preemptive Priority Scheduling
Process ID        Arrival Time    Burst Time      Priority        Completion Time          Turn Around Time         Waiting Time
    1                   0             4               2                 12                       12                      8
    2                   1             3               0                 4                        3                       0
    3                   2             1               3                 13                       11                      10
    4                   3             5               1                 9                        6                       1
    5                   4             2               4                 15                       11                      9
Average Turn Around Time : 8.600000
Average Wait Time : 5.600000
PS C:\Users\Archit\Desktop\cprog> 
```

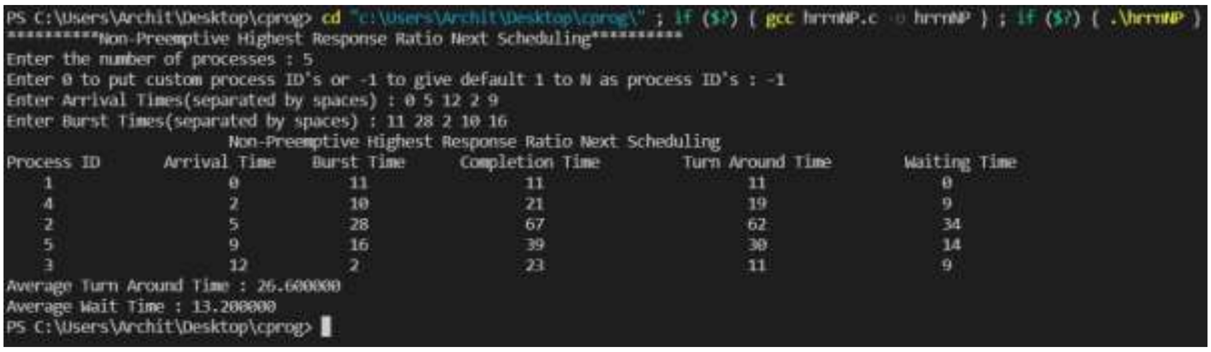
Highest Response Ration Next Scheduling (Non-Preemptive)

Code

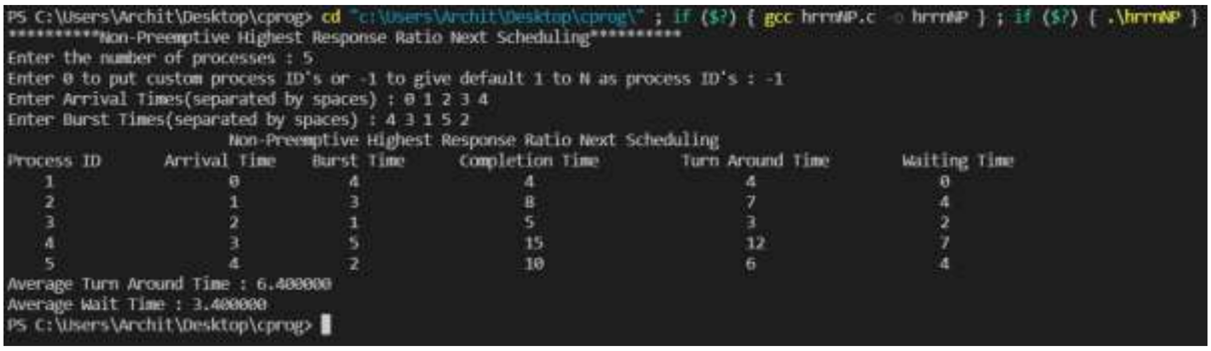
```
/*
Name: Archit Agrawal
Student ID: 202051213

The following code is a simulation of the  Non-Preemptive Highest Resoponse
Ratio Next Scheduling
Documentation:
The matrix mat stores all the data of the processes in the following code
    At index 0 -> Process ID
    At index 1 -> Arrival Time
    At index 2 -> Burst Time
    At index 3 -> finish or completion time
    At index 4 -> turn around time
    At index 5 -> waiting time

*/


#include<stdio.h>

//function to arrange processes by their arrival times
void arrangeArrival(int num, int mat[][6]);
//function to calculate completion time for each process
void completionTime(int num, int mat[][6]);
//function to calculate TAT for each process
void turnAroundTime(int num, int mat[][6]);
//function to calculate WT for each process
void waitingTime(int num, int mat[][6]);
//function to calculate average TAT
double avgTAT(int num, int mat[][6]);
//function to calculate average WT
double avgWT(int num, int mat[][6]);
//scheduling function that the user need to call
void HRRNScheduling(int num, int mat[][6]);

//utility function to print the result in formatted way
void printResult(int num, int mat[][6]);

void arrangeArrival(int num, int mat[][6]){
    for (int i = 0; i < num; i++) {
        for (int j = 0; j < num - i - 1; j++) {
            if (mat[j][1] > mat[j + 1][1]) {
                for (int k = 0; k < 3; k++) {

                    int temp = mat[j][k];
                    mat[j][k] = mat[j + 1][k];
                    mat[j + 1][k] = temp;
                }
            }
        }
    }

}

void completionTime(int num, int mat[][6]){
    int arr[num];
    for(int i = 0; i < num; i++) arr[i] = 0;
    int j = 0;
    int currTime = mat[0][1];
    int completed = 0;

    double hrrn[num];
    for(int i = 0; i < num; i++) hrrn[i] = 0.0;

    while(completed != num){
        while(j < num && arr[j] != -1 && mat[j][1] <= currTime){
            arr[j] = 1;
            j++;
        }

        for(int k = 0; k < num; k++){
            if(arr[k] == 1){
                hrrn[k] = 1.0 + (double)(currTime - mat[k][1])/mat[k][2];
            }
        }

        //for(int s = 0; s <num; s++) printf("%d, ", arr[s]);
        //printf("currTime : %d, j = %d, completed = %d\n", currTime, j,
completed);

        int currProc = 0;
        for(int k = 0; k < num; k++){
            if(arr[k] == 1){
                currProc = k;
                break;
            }
        }

        for(int k = currProc + 1; k < num; k++){
            if(arr[k] == 1 && hrrn[k] > hrrn[currProc]){
                currProc = k;
            }
        }

        currTime += mat[currProc][2];
        mat[currProc][3] = currTime;
        completed++;
        arr[currProc] = -1;
        hrrn[currProc] = -1.0;

        //printf("currTime : %d, j = %d, completed = %d\n", currTime, j,
completed);
        if(j < num && j - completed < 1 && mat[j][1] > currTime){
            currTime = mat[j][1];
        }


    }
    //printf("Completion Time Done");
    turnAroundTime(num, mat);
    //printf("TAT Done");
    waitingTime(num, mat);
    //printf("WT Done");
    printResult(num, mat);
}

void turnAroundTime(int num, int mat[][6]){
    for(int i = 0; i < num; i++) mat[i][4] = mat[i][3] - mat[i][1];
}

void waitingTime(int num, int mat[][6]){
    for(int i = 0; i < num; i++) mat[i][5] = mat[i][4] - mat[i][2];
}

double avgTAT(int num, int mat[][6]){
    int sum = 0;
    for(int i = 0; i < num; i++) sum += mat[i][4];
    return (double)sum/num;
}

double avgWT(int num, int mat[][6]){
    int sum = 0;
    for(int i = 0; i < num; i++) sum += mat[i][5];
```

```c
    return (double)sum/num;
}

void HRRNScheduling(int num, int mat[][6]){
    arrangeArrival(num, mat);
    completionTime(num, mat);
}

void printResult(int num, int mat[][6]){
    printf("\t\t\tNon-Preemptive Highest Response Ratio Next Scheduling\n");
    printf("Process ID \t Arrival Time \t Burst Time \t Completion Time \t 
Turn Around Time \t Waiting Time\n");
    for(int i = 0; i < num; i++){
        printf("    %d\t\t\t%d\t     %d   \t\t%d\t\t\t%d\t\t     %d\n", 
mat[i][0], mat[i][1], mat[i][2], mat[i][3], mat[i][4], mat[i][5]);
    }

    printf("Average Turn Around Time : %f\n", avgTAT(num, mat));
    printf("Average Wait Time : %f\n", avgWT(num, mat));
}

int main(){
    printf("**********Non-Preemptive Highest Response Ratio Next 
Scheduling**********\n");
    int n;
    printf("Enter the number of processes : ");
    scanf("%d", &n);
    int mat[n][6];

    printf("Enter 0 to put custom process ID's or -1 to give default 1 to N as 
process ID's : ");
    int id;
    scanf("%d", &id);

    if(id == -1){
        for(int i = 0; i < n; i++) mat[i][0] = i + 1;
    } else {
        printf("Enter Process ID's(separated by spaces) : ");
        for(int i = 0; i < n; i++){
            int s;
            scanf("%d", &s);
            mat[i][0] = s;
        }
    }

    printf("Enter Arrival Times(separated by spaces) : ");
    for(int i = 0; i < n; i++){
        int s;
        scanf("%d", &s);
        mat[i][1] = s;
    }

    printf("Enter Burst Times(separated by spaces) : ");

    for(int i = 0; i < n; i++){
        int s;
        scanf("%d", &s);
        mat[i][2] = s;
    }

    HRRNScheduling(n, mat);
}
```

Output



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { gcc hrrnNP.c -o hrrnNP } ; if ($?) { .\hrrnNP }
**********Non-Preemptive Highest Response Ratio Next Scheduling**********
Enter the number of processes : 5
Enter 0 to put custom process ID's or -1 to give default 1 to N as process ID's : -1
Enter Arrival Times(separated by spaces) : 0 5 12 2 9
Enter Burst Times(separated by spaces) : 11 28 2 10 16
                      Non-Preemptive Highest Response Ratio Next Scheduling
Process ID       Arrival Time    Burst Time      Completion Time         Turn Around Time        Waiting Time
    1                  0             11               11                      11                   0
    4                  2             10               21                      19                   9
    2                  5             28               67                      62                   34
    5                  9             16               39                      30                   14
    3                  12            2                23                      11                   9
Average Turn Around Time : 26.600000
Average Wait Time : 13.200000
PS C:\Users\Archit\Desktop\cprog>
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { gcc hrrnNP.c -o hrrnNP } ; if ($?) { .\hrrnNP }
**********Non-Preemptive Highest Response Ratio Next Scheduling**********
Enter the number of processes : 5
Enter 0 to put custom process ID's or -1 to give default 1 to N as process ID's : -1
Enter Arrival Times(separated by spaces) : 0 1 2 3 4
Enter Burst Times(separated by spaces) : 4 3 1 5 2
                      Non-Preemptive Highest Response Ratio Next Scheduling
Process ID       Arrival Time    Burst Time      Completion Time         Turn Around Time        Waiting Time
    1                  0             4                4                       4                    0
    2                  1             3                8                       7                    4
    3                  2             1                5                       3                    2
    4                  3             5                15                      12                   7
    5                  4             2                10                      6                    4
Average Turn Around Time : 6.400000
Average Wait Time : 3.400000
PS C:\Users\Archit\Desktop\cprog>
```

Highest Response Ration Next Scheduling (Preemptive)

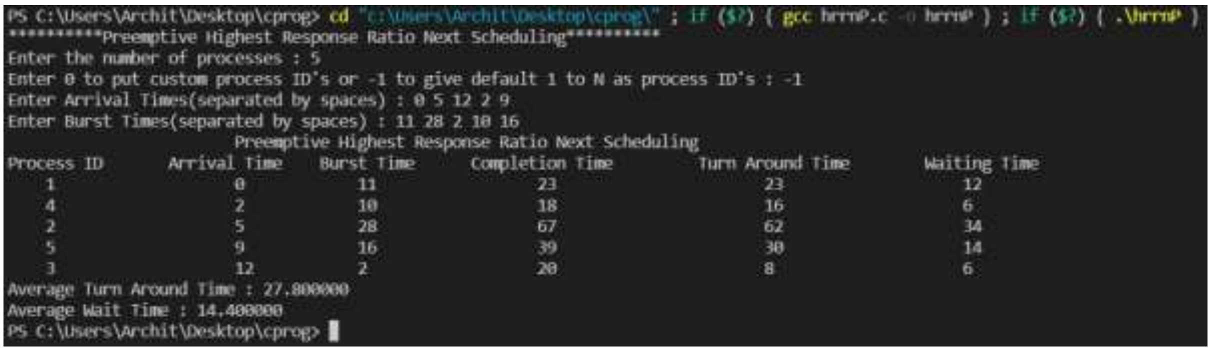
Code

```
/*
Name: Archit Agrawal
Student ID: 202051213

The following code is a simulation of the  Preemptive Highest Resoponse Ratio
Next Scheduling
Documentation:
The matrix mat stores all the data of the processes in the following code
    At index 0 -> Process ID
    At index 1 -> Arrival Time
    At index 2 -> Burst Time
    At index 3 -> finish or completion time
    At index 4 -> turn around time
    At index 5 -> waiting time

*/

#include<stdio.h>

//function to arrange processes by their arrival times
void arrangeArrival(int num, int mat[][6]);
//function to calculate completion time for each process
void completionTime(int num, int mat[][6]);
//function to calculate TAT for each process
void turnAroundTime(int num, int mat[][6]);
//function to calculate WT for each process
void waitingTime(int num, int mat[][6]);
//function to calculate average TAT
double avgTAT(int num, int mat[][6]);
//function to calculate average WT
double avgWT(int num, int mat[][6]);
//scheduling function that the user need to call
void HRRNScheduling(int num, int mat[][6]);

//utility function to print the result in formatted way
void printResult(int num, int mat[][6]);

void arrangeArrival(int num, int mat[][6]){
    for (int i = 0; i < num; i++) {
        for (int j = 0; j < num - i - 1; j++) {
            if (mat[j][1] > mat[j + 1][1]) {
                for (int k = 0; k < 3; k++) {

                    int temp = mat[j][k];
                    mat[j][k] = mat[j + 1][k];
                    mat[j + 1][k] = temp;
                }
            }
        }
    }

}

void completionTime(int num, int mat[][6]){
    int arr[num];
    for(int i = 0; i < num; i++) arr[i] = 0;
    int rem[num];
    for(int i = 0; i < num; i++) rem[i] = mat[i][2];
    int j = 0;
    int currTime = mat[0][1];
    int completed = 0;

    double hrrn[num];
    for(int i = 0; i < num; i++) hrrn[i] = 0.0;

    while(completed != num){

        while(j < num && arr[j] != -1 && mat[j][1] <= currTime){
            arr[j] = 1;
            mat[j][5] = 0;
            j++;
        }


        /****************************change*/
        for(int k = 0; k < num; k++){
            if(arr[k] == 1){
                hrrn[k] = 1.0 + (double)(mat[k][5])/mat[k][2];
            }
        }

        //for(int s = 0; s <num; s++) printf("%d, ", arr[s]);
        //printf("currTime : %d, j = %d, completed = %d\n", currTime, j,
completed);
```

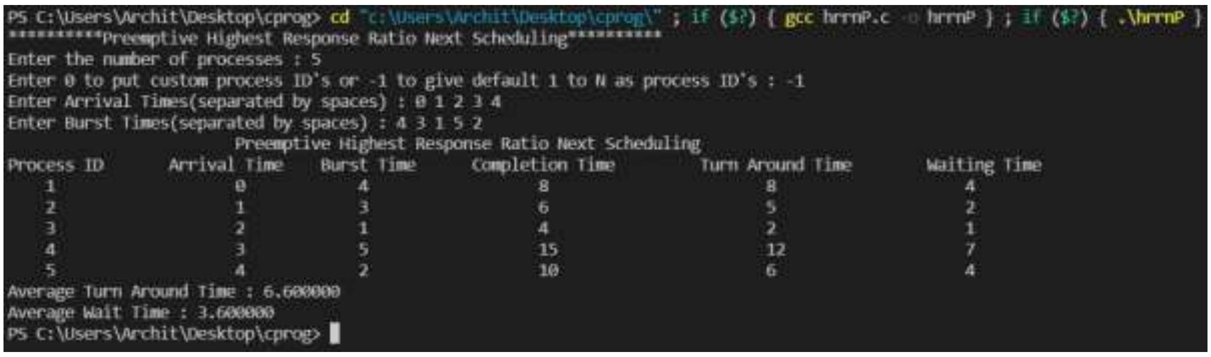
```
        int currProc = 0;
        for(int k = 0; k < num; k++){
            if(arr[k] == 1){
                currProc = k;
                break;
            }
        }

        for(int k = currProc + 1; k < num; k++){
            if(arr[k] == 1 && hrrn[k] > hrrn[currProc]){
                currProc = k;
            }
        }

        int duration = 0;

        if(j < num && mat[j][1] - currTime >= rem[currProc]){
            duration = mat[j][1] - currTime;
            currTime += rem[currProc];
            rem[currProc] = -1;
            completed++;
            mat[currProc][3] = currTime;
            arr[currProc] = -1;
        } else if(j < num && mat[j][1] - currTime < rem[currProc]){
            duration = mat[j][1] - currTime;
            rem[currProc] = rem[currProc] - (mat[j][1] - currTime);
            currTime =  mat[j][1];
        } else{
            duration = rem[currProc];
            currTime += rem[currProc];
            rem[currProc] = -1;
            completed++;
            mat[currProc][3] = currTime;
            arr[currProc] = -1;
        }

        //update wait time of each process other than the process that ran
        for(int k = 0; k < num; k++){
            if(k != currProc && arr[k] == 1){
                mat[k][5] += duration;
            }
        }

        if(j < num && j - completed < 1 && mat[j][1] > currTime){
            currTime = mat[j][1];
        }
    }
```

```c
    //printf("Completion Time Done");
    turnAroundTime(num, mat);
    //printf("TAT Done");
    waitingTime(num, mat);
    //printf("WT Done");
    printResult(num, mat);
}

void turnAroundTime(int num, int mat[][6]){
    for(int i = 0; i < num; i++) mat[i][4] = mat[i][3] - mat[i][1];
}

void waitingTime(int num, int mat[][6]){
    for(int i = 0; i < num; i++) mat[i][5] = mat[i][4] - mat[i][2];
}

double avgTAT(int num, int mat[][6]){
    int sum = 0;
    for(int i = 0; i < num; i++) sum += mat[i][4];
    return (double)sum/num;
}

double avgWT(int num, int mat[][6]){
    int sum = 0;
    for(int i = 0; i < num; i++) sum += mat[i][5];
    return (double)sum/num;
}

void HRRNScheduling(int num, int mat[][6]){
    arrangeArrival(num, mat);
    completionTime(num, mat);
}

void printResult(int num, int mat[][6]){
    printf("\t\t\tPreemptive Highest Response Ratio Next Scheduling\n");
    printf("Process ID \t Arrival Time \t Burst Time \t Completion Time \t
Turn Around Time \t Waiting Time\n");
    for(int i = 0; i < num; i++){
        printf("    %d\t\t\t%d\t     %d    \t\t%d\t\t\t%d\t\t     %d\n",
mat[i][0], mat[i][1], mat[i][2], mat[i][3], mat[i][4], mat[i][5]);
    }

    printf("Average Turn Around Time : %f\n", avgTAT(num, mat));
    printf("Average Wait Time : %f\n", avgWT(num, mat));
}

int main(){
    printf("**********Preemptive Highest Response Ratio Next Scheduling**********\n");
    int n;
    printf("Enter the number of processes : ");
    scanf("%d", &n);
    int mat[n][6];

    printf("Enter 0 to put custom process ID's or -1 to give default 1 to N as process ID's : ");
    int id;
    scanf("%d", &id);

    if(id == -1){
        for(int i = 0; i < n; i++) mat[i][0] = i + 1;
    } else {
        printf("Enter Process ID's(separated by spaces) : ");
        for(int i = 0; i < n; i++){
            int s;
            scanf("%d", &s);
            mat[i][0] = s;
        }
    }

    printf("Enter Arrival Times(separated by spaces) : ");

    for(int i = 0; i < n; i++){
        int s;
        scanf("%d", &s);
        mat[i][1] = s;
    }

    printf("Enter Burst Times(separated by spaces) : ");

    for(int i = 0; i < n; i++){
        int s;
        scanf("%d", &s);
        mat[i][2] = s;
    }

    HRRNScheduling(n, mat);
}
```

Output



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { gcc hrrnP.c -o hrrnP } ; if ($?) { .\hrrnP }
**********Preemptive Highest Response Ratio Next Scheduling**********
Enter the number of processes : 5
Enter 0 to put custom process ID's or -1 to give default 1 to N as process ID's : -1
Enter Arrival Times(separated by spaces) : 0 5 12 2 9
Enter Burst Times(separated by spaces) : 11 28 2 10 16
                        Preemptive Highest Response Ratio Next Scheduling
Process ID        Arrival Time    Burst Time      Completion Time         Turn Around Time        Waiting Time
    1                   0           11                  23                      23                   12
    4                   2           10                  18                      16                   6
    2                   5           28                  67                      62                   34
    5                   9           16                  39                      30                   14
    3                   12          2                   20                      8                    6
Average Turn Around Time : 27.800000
Average Wait Time : 14.400000
PS C:\Users\Archit\Desktop\cprog> 
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { gcc hrrnP.c -o hrrnP } ; if ($?) { .\hrrnP }
**********Preemptive Highest Response Ratio Next Scheduling**********
Enter the number of processes : 5
Enter 0 to put custom process ID's or -1 to give default 1 to N as process ID's : -1
Enter Arrival Times(separated by spaces) : 0 1 2 3 4
Enter Burst Times(separated by spaces) : 4 3 1 5 2
                        Preemptive Highest Response Ratio Next Scheduling
Process ID        Arrival Time    Burst Time      Completion Time         Turn Around Time        Waiting Time
    1                   0           4                   8                       8                    4
    2                   1           3                   6                       5                    2
    3                   2           1                   4                       2                    1
    4                   3           5                   15                      12                   7
    5                   4           2                   10                      6                    4
Average Turn Around Time : 6.600000
Average Wait Time : 3.600000
PS C:\Users\Archit\Desktop\cprog> 
```