Name: Shriom G Tripathi
Student ID: 11609674
Email Address: shriomtripathi33@gmail.com
GitHub Link: https://www.github.com/jay006/OS-Project
Code: Q2. (Non Pre-emptive priority scheduling)

Description:

I have applied a scheduling approach which is non pre-emptive similar to shortest job next in nature. The priority of each job is dependent on its estimated run time, and also the amount of time it has spent waiting. Jobs gain higher priority the longer they wait, which prevents indefinite postponement. The jobs that have spent a long time waiting compete against those estimated to have short run times. The priority can be computed as : Priority = 1+ Waiting time / Estimated run time

Algorithm:

I have used "Bubble Sort" algorithm to sort the processes based on the arrival time of precoess and updated there priority and sorted the order of process based on there current priority.

Steps of algorithm:

STEP 1: Take input of "Burst Time" and "Arrival Time" form user and sort the processes depending on there "Arrival Time" using **sortOnArrivalT(pros[])** .

STEP 2: Now start executing the process present in **procsses_arr[]** using the **startProcessing(pros[]).** In the method single process is executed and after execution of single process priority of each arrived process is updated using **updateProcesses(pros[],processTime,currentProcess)** .

STEP 3: After completion of STEP 2, print the order of execution of the processes along with there burst time and arrival time using **printProcess(pros[]).**

Name: Shriom G Tripathi
Student ID: 11609674
Email Address: shriomtripathi33@gmail.com
GitHub Link: https://www.github.com/jay006/OS-Project
Code: Q2. (Non Pre-emptive priority scheduling)

Bubble Sort(Sort Process on Priority):

```
int i , process_arr[tot_process]
loop i < process_arr.lengeth-1:
    int j = i+1
    loop j < processarr.length:
        if(processarr[i].priority < processarr[j].priority):
            struct process a = processarr[j];
            processarr[j] = processarr[i];
            processarr[i] = a;
```

Bubble Sort(Sort Process on Arrival Time):

```
int i , process_arr[tot_process]
loop i < process_arr.lengeth-1:
    int j = i+1
    loop j < processarr.length:
        if(pros[i].arvTime > pros[j].arvTime):
            struct process a = pros[j];
            pros[j] = pros[i];
            pros[i] = a;
```

Complexity:

In the this project I've used bubble sort to sort processes on arrival time time and on priority.

Name: Shriom G Tripathi
Student ID: 11609674
Email Address: shriomtripathi33@gmail.com
GitHub Link: https://www.github.com/jay006/OS-Project
Code: Q2. (Non Pre-emptive priority scheduling)

```
//sorting on arrival time.
        for(int i=0; i<no_process-1; i++){
                for(int j=i+1; j<no_process; j++)
                        if(pros[i].arvTime > pros[j].arvTime){
                                struct process a = pros[j];
                                pros[j] = pros[i];
                                pros[i] = a;
                        }}}
```

i.The above sorting snippet have $O(n^2)$ worst case compexity.

```
//sort process in decending order on priority.
        for(int i=currentProcess; i<no_process-1; i++){
                for(int j=i+1; j<no_process; j++){
                        if(pros[i].priority < pros[j].priority){
                                struct process a = pros[j];
                                pros[j] = pros[i];
                                pros[i] = a;}}}
```

ii. The above sorting snippet have $O(n^2)$ worst case compexity.

```
//start Processing
while(currentProcess < no_process){
        struct process topProcess = pros[currentProcess++];
        processTime =+ topProcess.burstTime;
        updateProcesses(pros,processTime,currentProcess);}
```

iii. The above loop have O(n) worst case compexity.

The total complexity of program= $O(n^2) + O(n^2) + O(n)$

$=O(n^2)$

Name: Shriom G Tripathi
Student ID: 11609674
Email Address: shriomtripathi33@gmail.com
GitHub Link: https://www.github.com/jay006/OS-Project
Code: Q2. (Non Pre-emptive priority scheduling)

Constraints:

In program constraints are applied on input i.e noOfProcess and on arrivalTime and burstTime.

NoOfProcess

```
printf("Enter the no. of Process\n");
scanf("%d",&no_process);
while(no_process < 1){
        printf("Please enter the number of process greater than one.\n");
        scanf("%d",&no_process);
} //no of process should not be less than one.
```

ArrivalTime.

```
while(pros[i].arvTime > -1){
        printf("Arrival Time : ");
        scanf("%d",&pros[i].arvTime);
        if(pros[i].arvTime > -1)
                break;
} //arrival time should not be less than zero
```

BurstTime

```
while(pros[i].burstTime > -1){
        printf("Burst Time : ");
        scanf("%d",&pros[i].burstTime);
        if(pros[i].burstTime > -1)
                break;
}//burst time should not be less than zero.
```

Name: Shriom G Tripathi
Student ID: 11609674
Email Address: shriomtripathi33@gmail.com
GitHub Link: https://www.github.com/jay006/OS-Project
Code: Q2. (Non Pre-emptive priority scheduling)

TestCases:

Input:

5

0 22

1 45

5 11

8 1

3 1

Output:

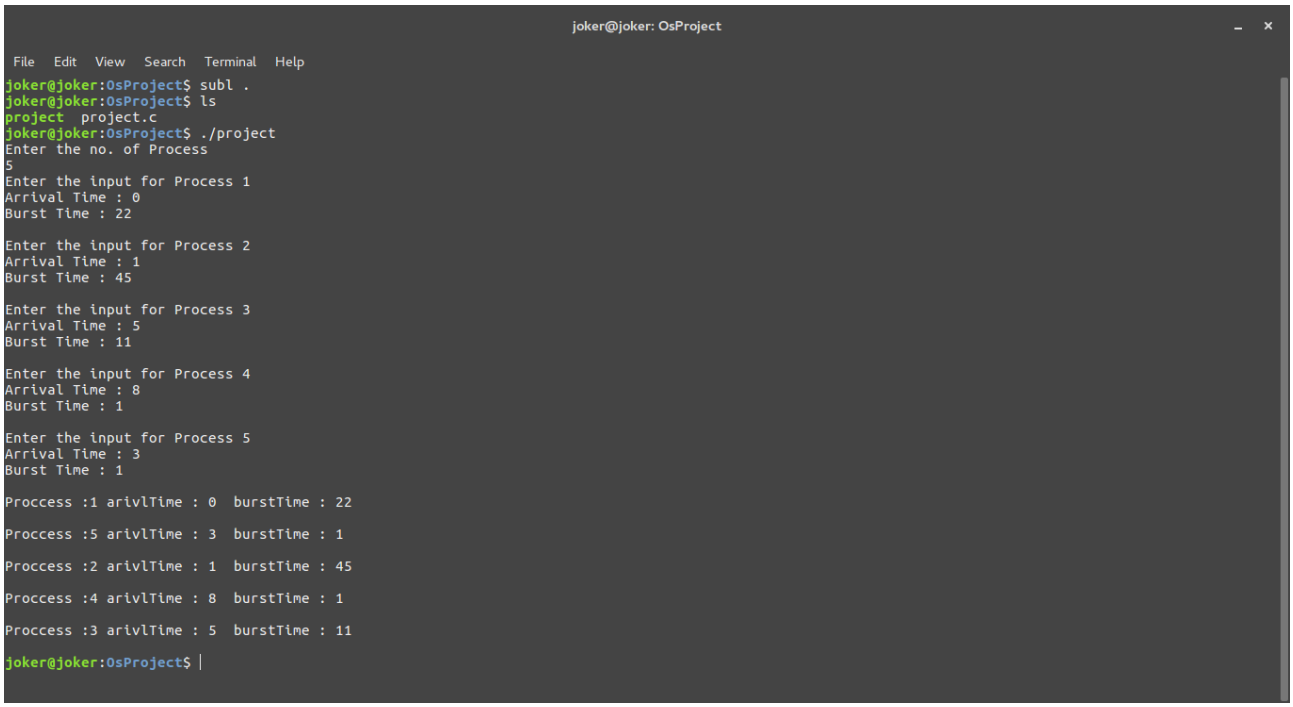
Proccess :1 arivlTime : 0 burstTime : 22

Proccess :5 arivlTime : 3 burstTime : 1

Proccess :2 arivlTime : 1 burstTime : 45

Proccess :4 arivlTime : 8 burstTime : 1

Proccess :3 arivlTime : 5 burstTime : 11



Name: Shriom G Tripathi
Student ID: 11609674
Email Address: shriomtripathi33@gmail.com
GitHub Link: https://www.github.com/jay006/OS-Project
Code: Q2. (Non Pre-emptive priority scheduling)

First line of input is for “noOfProcesses” n, the subsequient n lines takes input as arrival time and burst time. Output show the order of execution of the the processes priority as it changes by given equation Priority = 1 + waitingTime/executionTime.

Code:

```
#include<stdio.h>

int processTime = 0;                    //->stores the current execution time.
int no_process = 0;                             //->no of process

struct process{
        int id;
        int arvTime ;
        int burstTime ;
        int waitTime ;
        float priority ;
};

void sortOnArrivalT(struct process pros[]);
void startProcessing(struct process pros[]);
void updateProcesses(struct process pros[], int processTime, int currentProcess);
void printProcess(struct process pros[]);

int main(){
        printf("Enter the no. of Process\n");

        scanf("%d",&no_process);
```

Name: Shriom G Tripathi
Student ID: 11609674
Email Address: shriomtripathi33@gmail.com
GitHub Link: https://www.github.com/jay006/OS-Project
Code: Q2. (Non Pre-emptive priority scheduling)

```
while(no_process < 1){
        printf("Please enter the number of process greater than one.\n");
        scanf("%d",&no_process);
}

//init the sturct.
struct process pros[no_process];

//takeing input 'arvTime' and 'burstTime' for all the precesses.
for(int i=0; i<no_process; i++){

        printf("Enter the input for Process %d\n", i+1 );
        pros[i].id = i+1;
        pros[i].waitTime = 0;
        pros[i].priority = 1;
        //arvTime
        pros[i].arvTime = 0;
        while(pros[i].arvTime > -1){
                printf("Arrival Time : ");
                scanf("%d",&pros[i].arvTime);
                if(pros[i].arvTime > -1)
                        break;
        }
        //burstTime
        pros[i].burstTime = 0;
        while(pros[i].burstTime > -1){
                printf("Burst Time : ");
                scanf("%d",&pros[i].burstTime);
                if(pros[i].burstTime > -1)
```

Name: Shriom G Tripathi
Student ID: 11609674
Email Address: shriomtripathi33@gmail.com
GitHub Link: https://www.github.com/jay006/OS-Project
Code: Q2. (Non Pre-emptive priority scheduling)

```
                        break;
                }
                printf("\n");
        }
        sortOnArrivalT(pros);
        startProcessing(pros);
        printProcess(pros);
        return 0;
}

//method to sort the struct depending on arival time and burst time.
void sortOnArrivalT(struct process pros[]){
        //sorting on arrival time.
        for(int i=0; i<no_process-1; i++){
                for(int j=i+1; j<no_process; j++){
                        if(pros[i].arvTime > pros[j].arvTime){
                                struct process a = pros[j];
                                pros[j] = pros[i];
                                pros[i] = a;
                        }
                }
        }
}

//process algo.
void startProcessing(struct process pros[]){
        int currentProcess = 0;      //process index starts for 0
        //start Processing
        while(currentProcess < no_process){
```

Name: Shriom G Tripathi
Student ID: 11609674
Email Address: shriomtripathi33@gmail.com
GitHub Link: https://www.github.com/jay006/OS-Project
Code: Q2. (Non Pre-emptive priority scheduling)

```
                struct process topProcess = pros[currentProcess++];
                processTime =+ topProcess.burstTime;
                updateProcesses(pros,processTime,currentProcess);
        }
}
//method to update process queue 'pors[]'
void updateProcesses(struct process pros[], int processTime, int currentProcess){
        //update the priority of each process depenting on waitTime and execution time.
        for(int i = currentProcess; i<no_process; i++){
                float waitTime = processTime - pros[i].arvTime;
                pros[i].priority = 1   + waitTime/(float)pros[i].burstTime;
        }
        //sort process in decending order on priority.
        for(int i=currentProcess; i<no_process-1; i++){
                for(int j=i+1; j<no_process; j++){
                        if(pros[i].priority < pros[j].priority){
                                struct process a = pros[j];
                                pros[j] = pros[i];
                                pros[i] = a;
                        }
                }
        }
}
//method to pring the process.
void printProcess(struct process pros[]){
        for(int i=0; i<no_process; i++){
                printf("Proccess :%d arivlTime : %d   burstTime : %d\n\n", pros[i].id, pros[i].arvTime,  pros[i].burstTime);
```

Name: Shriom G Tripathi
Student ID: 11609674
Email Address: shriomtripathi33@gmail.com
GitHub Link: https://www.github.com/jay006/OS-Project
Code: Q2. (Non Pre-emptive priority scheduling)

```
        }
}
```