

Name: Nguyễn Công Phi

ID: 19522006

Class: KTPM2019 / IT007.L11

OPERATING SYSTEM LAB 4'S REPORT

SUMMARY

Task			Status	Page
Section 4.5	Task 1	Code và giải thích	Done	2
		Chạy trong ubuntu	Done	8
	Task 2	Code và giải thích	Done	9
		Chạy trong ubuntu	Done	20
	Task 3	Code và giải thích	Done	21
		Chạy trong ubuntu	Done	33

Self-scores: 3/3

Section 4.5

1. Task 1: Viết chương trình mô phỏng giải thuật SJF với các yêu cầu sau:

- ❖ Nhập số lượng process
- ❖ Nhập process name, arrival time, burst time
- ❖ In ra Process name, response time, waiting time, turnaround time, average waiting time, average turnaround time

- Code của giải thuật (Code được viết bằng ngôn ngữ C++) :

```
- #include <iostream>
- #include <string>
- using namespace std;
- class Process
- {
- public:
-     int pname = 0;
-     float paTime = 0.f, pbTime = 0.f;
-     float wTime = 0.f, taTime = 0.f;
-     float staTime = -1.f, fiTime = 0.f, resTime = 0.f;
-     //Function
-     void Nhap() {
-         cout << "ProcessName, Process Arrival Time, Process Burst Time: ";
-         cin >> pname >> paTime >> pbTime;
-     }
-     void Xuat(){
-         printf("%d\t%8.1f\t%8.1f\t%8.1f\t%8.1f\t%8.1f\t%8.1f",pname,paTime,pbTime,staTime, taTime,fiTime,wTime,resTime);
-         cout<<"\n";
-     }
- };
```

- Class Process chứa các thuộc tính và các phương thức nhập xuất các tiến trình

```
class Queue
{
public:
    Process* p = NULL;
    int soLuong = 0;
    Queue(){
    }
    ~Queue(){
        if (soLuong != 0)
```

```

        delete p;
    }
    void Push(Process temp){
        Process* t = new Process[soLuong];
        for (int i = 0; i < soLuong; i++)
        {
            t[i] = p[i];
        }
        soLuong += 1;
        p = new Process[soLuong];
        for (int i = 0; i < soLuong - 1; i++)
        {
            p[i] = t[i];
        }
        p[soLuong - 1] = temp;
        delete t;
    }
    Process PopAtMin(){
        float bTimeMin = p[0].pbTime;
        for (int i = 1; i < soLuong; i++)
        {
            if (p[i].pbTime < bTimeMin)
            {
                bTimeMin = p[i].pbTime;
            }
        }
        Process res;
        int index = 0;
        for (int i = 0; i < soLuong; i++)
        {
            if (p[i].pbTime == bTimeMin)
            {
                res = p[i];
                index = i;
                break;
            }
        }
        Process* t = new Process[soLuong - 1];
        for (int i = 0; i < index; i++)
        {
            t[i] = p[i];
        }
        for (int i = index + 1; i < soLuong; i++)
        {
            t[i - 1] = p[i];
        }
    }

```

```

    soLuong -= 1;
    p = new Process[soLuong];
    for (int i = 0; i < soLuong; i++)
    {
        p[i] = t[i];
    }
    delete t;
    return res;
}
bool isEmpty(){
    if (soLuong == 0) return true;
    else return false;
}
};

```

- Class Queue là 1 queue chứa các Process với hàm:
 - o Push: để thêm process vào Queue
 - o PopAtMin: để lấy ra tiến trình có burst nhỏ nhất có trong queue và lấy ra khỏi Queue
 - o isEmpty: Một bool để xét xem có process nào trong Queue không

```

- class Algorithm
- {
- public:
-     Process pc[100];
-     int soLuong = 0;
-     float avgWTime = 0.f, avgTATime = 0.f;
-     void Nhap() {
-         cout << "Nhap so tien trinh: ";
-         int n; cin >> n;
-         soLuong = n;
-         for (int i = 0; i < n; i++)
-         {
-             pc[i].Nhap();
-         }
-     }
-     void Xuat() {
-         cout << "pName    ArrTime    BurstTime    Start    TAT    Fini
sh         Waiting    Res\n";
-         for (int i = 0; i < soLuong; i++)
-         {
-             pc[i].Xuat();
-         }
-     }
- }

```

```

-     }
-     void ShortJobFirst(){
-         //cout << "pName   ArrTime   BurstTime           Start           TAT           Fi
nish\n";
-         Queue pQueue;
-         Process* Ketqua = new Process[soLuong];
-         //Sắp xếp các tiến trình theo arrival time
-         for (int i = 0; i < soLuong - 1; i++)
-         {
-             for (int j = i + 1; j < soLuong; j++)
-             {
-                 if (pc[i].paTime > pc[j].paTime)
-                 {
-                     Process temp = pc[i];
-                     pc[i] = pc[j];
-                     pc[j] = temp;
-                 }
-             }
-         }
-         float time = pc[0].paTime;
-         int so = soLuong;
-         int index = 0;
-         for (int i = 0; i < so; i++)
-         {
-             for (int j = index; j < soLuong; j++)
-             {
-                 if (pc[j].paTime <= time)
-                 {
-                     pQueue.Push(pc[j]);
-                 }
-                 else
-                 {
-                     index = j;
-                     break;
-                 }
-             }
-             if (j == soLuong - 1)
-             {
-                 index = soLuong;
-             }
-         }

```

```

-     }
-     Process k;
-     if (pQueue.isEmpty()==true)
-     {
-         i--;
-         time = pc[index].paTime;
-         for (int j = index; j < soLuong; j++)
-         {
-             if (pc[j].paTime <= time)
-             {
-                 pQueue.Push(pc[j]);
-             }
-             else
-             {
-                 index = j;
-                 break;
-             }
-             if (j == soLuong - 1)
-             {
-                 index = soLuong;
-             }
-         }
-         continue;
-     }
-     else k = pQueue.PopAtMin();
-     k.staTime = time;
-     k.taTime = k.pbTime + time - k.paTime;
-     time += k.pbTime;
-     k.fiTime = time;
-     k.wTime = k.staTime - k.paTime;
-     k.resTime=k.staTime-k.paTime;
-     Ketqua[i] = k;
-     //Ketqua[i].Xuat();
- }
- for (int i = 0; i < soLuong; i++)
- {
-     pc[i] = Ketqua[i];
- }
- delete Ketqua;

```

```

- //Tính avgWaiting và avgTAT
- for (int i = 0; i < soLuong; i++)
- {
-     avgWTime += pc[i].wTime;
-     avgTATime += pc[i].taTime;
- }
- avgWTime /= soLuong;
- avgTATime /= soLuong;
- cout << "\nShort Job First:\nAverage Waiting Time: " << avgWTime << "\nAverage
Turn Around Time: " << avgTATime << "\n";
- }
- };
-

```

- Class Algorithm:

- Gồm hai hàm nhập xuất các tiến trình
- Hàm Short Job First là giải thuật Short Job First:
 - B1: Sắp xếp các tiến trình theo arrival time tăng dần
 - B2: Chọn thời gian hiện tại là thời gian của arrival time đầu tiên
 - B3: Xếp tất cả tiến trình có thời gian bằng với thời gian hiện tại vào Queue và lấy ra tiến trình có thời gian burst time bé nhỏ nhất với hàm PopAtMin để thực hiện, nếu có nhiều hơn một thì lấy tiến trình nằm trước trong Queue để thực hiện
 - B4: Thực hiện xong tiến trình thì sẽ lấy thời gian hiện tại bằng giá trị lớn hơn thời gian lúc hoàn thành tiến trình và thời gian lúc tiến trình tiếp theo tới
 - B5: Nếu trong hàng đợi còn tiến trình hoặc index nhỏ hơn số tiến trình thì sẽ tiếp tục với bước 3 còn không sẽ in ra danh sách các tiến trình được thực hiện và kết thúc

```

- int main()
- {
-     Algorithm a;
-     a.Nhap();
-     a.ShortJobFirst();
-     a.Xuat();
-     return 0;
- }

```

- Đoạn code chương trình chính:

- Tạo 1 object algorithm, nhập các process, thực hiện short job first và in ra các thứ tự thực hiện các tiến trình bằng hàm xuất

- Kết quả chạy trên Ubuntu:

- Testcase 1:

```

phi-19522006@phi19522006-VirtualBox: ~/Documents/LAB 4
./SJF
Nhap so tien trinh: 3
ProcessName, Process Arrival Time, Process Burst Time: 1 99 12
ProcessName, Process Arrival Time, Process Burst Time: 2 33 11
ProcessName, Process Arrival Time, Process Burst Time: 3 12 1

Short Job First:
Average Waiting Time: 0
Average Turn Around Time: 8
pName ArrTime BurstTime Start TAT Finish Waiting Res
3 12.0 1.0 12.0 1.0 13.0 0.0 0.0
2 33.0 11.0 33.0 11.0 44.0 0.0 0.0
1 99.0 12.0 99.0 12.0 111.0 0.0 0.0
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$

```

- - Với 3 tiến trình có thời gian arrival time chênh nhau lớn hơn burst time thì SJF sẽ thực hiện như FCFS

○ Testcase 2:

```

phi-19522006@phi19522006-VirtualBox: ~/Documents/LAB 4
./SJF
Nhap so tien trinh: 4
ProcessName, Process Arrival Time, Process Burst Time: 1 2 3
ProcessName, Process Arrival Time, Process Burst Time: 2 1 4
ProcessName, Process Arrival Time, Process Burst Time: 3 3 1
ProcessName, Process Arrival Time, Process Burst Time: 4 2 2

Short Job First:
Average Waiting Time: 3
Average Turn Around Time: 5.5
pName ArrTime BurstTime Start TAT Finish Waiting Res
2 1.0 4.0 1.0 4.0 5.0 0.0 0.0
3 3.0 1.0 5.0 3.0 6.0 2.0 2.0
4 2.0 2.0 6.0 6.0 8.0 4.0 4.0
1 2.0 3.0 8.0 9.0 11.0 6.0 6.0
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$

```

- - Với 4 tiến trình trên thì tiến trình 2 với thời gian đến đầu tiên sẽ được thực hiện trước, sau khi thực hiện xong thì đã là giây thứ 5, tất cả tiến trình khác đã vào hàng đợi nên nó sẽ thực hiện theo thứ tự tiến trình có burst time nhỏ nhất trước nên thứ tự thực hiện sẽ là 3 4 1

○ Testcase 3:


```

phi-19522006@phi19522006-VirtualBox: ~/Documents/LAB 4$ ./SJF
Nhap so tien trinh: 4
ProcessName, Process Arrival Time, Process Burst Time: 1 0 9
ProcessName, Process Arrival Time, Process Burst Time: 2 0 4
ProcessName, Process Arrival Time, Process Burst Time:
3 0 1
ProcessName, Process Arrival Time, Process Burst Time: 4 0 2

Short Job First:
Average Waiting Time: 2.75
Average Turn Around Time: 6.75
pName  ArrTime  BurstTime      Start      TAT      Finish      Waiting      Res
3       0.0      1.0           0.0         1.0        1.0         0.0         0.0
4       0.0      2.0           1.0         3.0        3.0         1.0         1.0
2       0.0      4.0           3.0         7.0        7.0         3.0         3.0
1       0.0      9.0           7.0        16.0       16.0        7.0         7.0
phi-19522006@phi19522006-VirtualBox: ~/Documents/LAB 4$

```

- Các tiến trình có cùng arrival time là 0 nên sẽ được thực thi theo thứ tự tiến trình burst time nhỏ nhất trước, các tiến trình có burst time lớn nhất sau

2. Task 2: Viết chương trình mô phỏng giải thuật SRT với các yêu cầu sau:

- ❖ Nhập số lượng process
- ❖ Nhập process name, arrival time, burst time 13
- ❖ In ra Process name, response time, waiting time, turnaround time, average waiting time, average turnaround time

- Code của thuật toán SRT(Được viết bằng C++)

```

- #include <iostream>
- #include <string>
- using namespace std;
-
- class Process
- {
- public:
-     int pname=0;
-     float paTime=0.f, pbTime=0.f;
-     float wTime = 0.f, taTime = 0.f;
-     float staTime = -1.f, fiTime=0.f;
-     float timeLeft = 0.f, resTime = 0.f; // use for RR and SRJF
-     //Function
-     void Nhap()
-     {
-         cout << "ProcessName, Process Arrival Time, Process Burst Time: ";
-         cin >> pname>> paTime>> pbTime;
-         timeLeft = pbTime;
-     }
-     void Xuat()

```

```

-     {
-         printf("%d      %8.1f\t%8.1f\t%8.1f\t%8.1f\t%8.1f\t%8.1f\t%8.1f",pname,paT
ime,pbTime,staTime, taTime,fiTime,wTime,timeLeft,resTime);
-         cout << "\n";
-     }
- };

```

- Class Process chứa các thuộc tính và các phương thức nhập xuất các tiến trình

```

- class Queue
- {
- public:
-     Process* p = NULL;
-     int soLuong = 0;
-     Queue()
-     {
-
-     }
-     ~Queue()
-     {
-         if(soLuong!=0)
-             delete p;
-     }
-     void Push(Process temp)
-     {
-         Process* t = new Process[soLuong];
-         for (int i = 0; i < soLuong; i++)
-         {
-             t[i] = p[i];
-         }
-         soLuong += 1;
-         p = new Process[soLuong];
-         for (int i = 0; i < soLuong-1; i++)
-         {
-             p[i] = t[i];
-         }
-         p[soLuong - 1] = temp;
-         delete t;
-     }
-     Process ViewAtMin()
-     {

```

```

-     float TimeMin = p[0].timeLeft;
-     for (int i = 1; i < soLuong; i++)
-     {
-         if (p[i].timeLeft < TimeMin)
-         {
-             TimeMin = p[i].pbTime;
-         }
-     }
-     Process res;
-     int index = 0;
-     for (int i = 0; i < soLuong; i++)
-     {
-         if (p[i].timeLeft == TimeMin)
-         {
-             res = p[i];
-             index = i;
-             break;
-         }
-     }
-     return res;
- }
- Process PopAtMin2()
- {
-     if (soLuong == 0)
-     {
-         Process error;
-         error.pname = -1;
-         return error;
-     }
-     else
-     {
-         float TimeMin = p[0].timeLeft;
-         for (int i = 1; i < soLuong; i++)
-         {
-             if (p[i].timeLeft < TimeMin)
-             {
-                 TimeMin = p[i].timeLeft;
-             }
-         }
-     }

```

```

-         Process res;
-         int index = 0;
-         for (int i = 0; i < soLuong; i++)
-         {
-             if (p[i].timeLeft == TimeMin)
-             {
-                 res = p[i];
-                 index = i;
-                 break;
-             }
-         }
-         Process* t = new Process[soLuong - 1];
-         for (int i = 0; i < index; i++)
-         {
-             t[i] = p[i];
-         }
-         for (int i = index + 1; i < soLuong; i++)
-         {
-             t[i - 1] = p[i];
-         }
-         soLuong -= 1;
-         p = new Process[soLuong];
-         for (int i = 0; i < soLuong; i++)
-         {
-             p[i] = t[i];
-         }
-         delete t;
-         return res;
-     }
- }
- bool isEmpty()
- {
-     if (soLuong == 0) return true;
-     else return false;
- }
- void ViewQueue()
- {
-     for (int i = 0; i < soLuong; i++)
-     {

```

```

-         printf("%d    %8.1f\t%8.1f\t%8.1f", p[i].pname, p[i].paTime, p[i].pbTime, p
[i].staTime);
-         cout << "\n";
-     }
-
- }
- Process PopAndMerge()
- {
-     Process first = p[0];
-     int indexoffirst=0;
-     for (int i = 1; i < soLuong; i++)
-     {
-         if (first.pname == p[i].pname)
-         {
-             first=p[i];
-             for (int j = indexoffirst; j < soLuong - 1; j++)
-             {
-                 p[j] = p[j + 1];
-             }
-             indexoffirst = i - 1;
-             soLuong -= 1;
-         }
-     }
-     for (int j = indexoffirst; j < soLuong - 1; j++)
-     {
-         p[j] = p[j + 1];
-     }
-     soLuong -= 1;
-     return first;
- }
- };

```

- Class Queue là 1 hàng đợi có chứa các Process và các phương thức
 - o Push: Thêm một process vào hàng đợi
 - o PopAtMin: Lấy process có timeLeft nhỏ nhất ra khỏi hàng đợi
 - o isEmpty: một bool xét thử Queue có đang trống hay không
 - o ViewQueue: một void dùng để xem những process đang có trong queue(em dùng nó trong quá trình debug)

- PopAndMerge: Nhóm tiến trình lại với nhau bởi khi push vào queue thì các tiến trình được push theo khoảng thời gian thực hiện, thời gian dừng thực hiện(có tiến trình khác chiếm chỗ hoặc do tiến trình đã được thực hiện xong)

```

- class Algorithm
- {
- public:
-     Process pc[100];
-     int soLuong=0;
-     float avgWTime = 0.f, avgTATime = 0.f;
-     void Nhap()
-     {
-         cout << "Nhap so tien trinh: ";
-         int n; cin >> n;
-         soLuong = n;
-         for (int i = 0; i < n; i++)
-         {
-             pc[i].Nhap();
-         }
-     }
-     void Xuat()
-     {
-         cout << "pName    ArrTime    BurstTime    Start    TAT    Fini
sh        Waiting    TimeLeft    Res\n";
-         for (int i = 0; i < soLuong; i++)
-         {
-             pc[i].Xuat();
-         }
-     }
-     void ShortRemainingFirst()
-     {
-         cout << "pName    ArrTime    BurstTime    Start    TAT    Fini
sh        Waiting    TimeLeft    Res\n";
-         //cout << "pName    ArrTime    BurstTime    Start    TAT    Fi
nish\n";
-         Queue pQueue;
-         Queue run;
-         //Sắp xếp các tiến trình theo arrival time
-         for (int i = 0; i < soLuong - 1; i++)
-         {
-             for (int j = i + 1; j < soLuong; j++)

```

```

-         {
-             if (pc[i].paTime > pc[j].paTime)
-             {
-                 Process temp = pc[i];
-                 pc[i] = pc[j];
-                 pc[j] = temp;
-             }
-         }
-     }
-
-     int index = 0;
-     float time = pc[0].paTime;
-     Process currentProcess;
-     currentProcess.timeLeft=-1;
-     //Push những process cùng arrival time vào trong queue
-     int j = index;
-     while (j + 1 < soLuong && pc[j].paTime == pc[j + 1].paTime)
-     {
-         j++;
-     }
-     for (int k = index; k <= j; k++)
-     {
-         pQueue.Push(pc[k]);
-     }
-     index = j + 1;
-     currentProcess = pQueue.PopAtMin2();
-     while(true)
-     {
-         //tất cả các tiến trình đã vào trong queue
-         if (index == soLuong)
-         {
-             //cout << "Index: " << index << " So luong: " << soLuong<<"\n";
-             while (pQueue.isEmpty() != true)
-             {
-                 if (currentProcess.timeLeft == -1)
-                     currentProcess = pQueue.PopAtMin2();
-                 if (currentProcess.staTime == -1)
-                 {
-                     currentProcess.staTime = time;
-                 }
-             }

```

```

-         time += currentProcess.timeLeft;
-         currentProcess.taTime = time - currentProcess.paTime;
-         currentProcess.fiTime = time;
-         currentProcess.timeLeft = 0;
-         currentProcess.wTime = currentProcess.fiTime - currentProcess.paTim
e - currentProcess.pbTime;
-         run.Push(currentProcess);
-         currentProcess.Xuat();
-         currentProcess.timeLeft = -1;
-     }
-     break;
- }
- else
- {
-     //Xét trường hợp đang chạy thì không bị process mới nhảy vào
-     while (pc[index].paTime - currentProcess.timeLeft > time)
-     {
-         if (currentProcess.timeLeft == -1)
-         {
-             if (pQueue.isEmpty() == true) break;
-             else currentProcess = pQueue.PopAtMin2();
-         }
-         //if (currentProcess.staTime == -1)
-         {
-             currentProcess.staTime = time;
-         }
-         time += currentProcess.timeLeft;
-         currentProcess.taTime = time - currentProcess.paTime;
-         currentProcess.timeLeft = 0;
-         currentProcess.fiTime = time;
-         currentProcess.wTime = currentProcess.fiTime - currentProcess.paTim
e - currentProcess.pbTime;
-         run.Push(currentProcess);
-         currentProcess.Xuat();
-         // cout << "In ra o while thu 2\n";
-         if (pQueue.isEmpty() != true)
-         {
-             currentProcess = pQueue.PopAtMin2();
-         }
-     }
- }

```



```

-         else
-         {
-             currentProcess.timeLeft = -1;
-         }
-     }
-     if (pQueue.isEmpty() == true && currentProcess.timeLeft== -1)
-     {
-         time = pc[index].paTime;
-         int j = index;
-         while (j + 1 < soLuong && pc[j].paTime == pc[j + 1].paTime)
-         {
-             j++;
-         }
-         for (int k = index; k <= j; k++)
-         {
-             pQueue.Push(pc[k]);
-         }
-         index = j + 1;
-         currentProcess = pQueue.PopAtMin2();
-     }
-     //Nếu như có 1 process mới nhảy vào hàng đợi lúc đang thực hiện
-     else
-     {
-         if (currentProcess.staTime == -1)
-         {
-             currentProcess.staTime = time;
-         }
-         currentProcess.timeLeft -= (pc[index].paTime - time);
-         //Push những process có arrival time vào trong queue
-         int j = index;
-         time = pc[j].paTime;
-         while (j + 1 < soLuong && pc[j].paTime == pc[j + 1].paTime)
-         {
-             j++;
-         }
-         for (int k = index; k <= j; k++)
-         {
-             pQueue.Push(pc[k]);
-         }
-     }

```

```

-         index = j + 1;
-
-         //Xem xét tiến trình đang thực hiện có thể bị đá ra hay không
-         Process maybeCurrent = pQueue.ViewAtMin();
-         if (currentProcess.timeLeft > maybeCurrent.timeLeft)
-         {
-             run.Push(currentProcess);
-             currentProcess.Xuat();
-             pQueue.Push(currentProcess);
-             //cout << currentProcess.pname << " bị đá bởi " << maybeCurrent
-             .pname<<"\n";
-
-             currentProcess = maybeCurrent;
-             pQueue.PopAtMin2();
-             time = maybeCurrent.paTime;
-         }
-         else time = pc[j].paTime;
-     }
-     if (index == soLuong)
-     {
-         while (pQueue.isEmpty() != true)
-         {
-             if (currentProcess.timeLeft == -1)
-                 currentProcess = pQueue.PopAtMin2();
-             if (currentProcess.staTime == -1)
-             {
-                 currentProcess.staTime = time;
-             }
-             time += currentProcess.timeLeft;
-             currentProcess.taTime = time - currentProcess.paTime;
-             currentProcess.fiTime = time;
-             currentProcess.timeLeft = 0;
-             currentProcess.wTime = currentProcess.fiTime - currentProcess.p
-             aTime - currentProcess.pbTime;
-             run.Push(currentProcess);
-             currentProcess.Xuat();
-             currentProcess.timeLeft = -1;
-         }
-         break;
-     }
- }

```

```

-     }
-
-     }
-     if (currentProcess.timeLeft != -1)
-     {
-         if (currentProcess.staTime == -1)
-         {
-             currentProcess.staTime = time;
-         }
-         time += currentProcess.timeLeft;
-         currentProcess.taTime = time - currentProcess.paTime;
-         currentProcess.fiTime = time;
-         currentProcess.timeLeft = 0;
-         currentProcess.wTime = currentProcess.fiTime - currentProcess.paTime - curr
entProcess.pbTime;
-         run.Push(currentProcess);
-         currentProcess.Xuat();
-         currentProcess.timeLeft = -1;
-     }
-     int i = 0;
-     while (run.isEmpty() != true)
-     {
-         pc[i]=run.PopAndMerge();
-         i++;
-     }
-     //Tính avgWaiting và avgTAT
-     for (int i = 0; i < soLuong; i++)
-     {
-         avgWTime += pc[i].wTime;
-         avgTATime += pc[i].taTime;
-         pc[i].resTime = pc[i].staTime - pc[i].paTime;
-     }
-     avgWTime /= soLuong;
-     avgTATime /= soLuong;
-     cout << "\nShort Remaining Job First:\nAverage Waiting Time: " << avgWTime << "
\nAverage Turn Around Time: " << avgTATime << "\n";
- }
- };

```

- Class Algorithm:

- Chứa các process và Queue để có thể Push các Process vào
- Và một hàm thực hiện thuật toán SRT
 - B1: Sắp xếp các process theo arrival time tăng dần
 - B2: Chọn time là arrival time của process đầu tiên. Thêm các process có cùng arrival time vào hàng đợi.
 - B3: Chọn process có timeleft nhỏ nhất trong queue để thực thi. Xét trong quá trình thực thi có process nào đến hay không.
 - Nếu không có thì tiếp tục thực hiện cho đến hết timeleft.
 - Nếu có thì xét xem timeleft của tiến trình vừa đến có nhỏ hơn timeleft của tiến trình hiện tại hay không, nếu có thì push tiến trình đang thực hiện vào trong queue và thực thi tiến trình có timeleft nhỏ nhất trong các tiến trình thỏa, thực thi lặp lại bước 3 cho đến khi tất cả các tiến trình vào trong queue
 - B4: Thực thi tất cả tiến trình trong queue theo thứ tự timeleft nhỏ nhất trước, timeleft lớn nhất sau.
 - B5: Kết thúc và xuất ra kết quả

```

- int main()
- {
-     Algorithm a;
-     a.Nhap();
-     a.ShortRemainingFirst();
-     a.Xuat();
- }

```

Đây là chương trình chính, chương trình sẽ yêu cầu nhập các process sau đó thực hiện thuật toán SRT và xuất ra kết quả

- Kết quả chạy trên ubuntu

- Testcase 1:

```

phi-19522006@phi19522006-VirtualBox: ~/Documents/LAB 4
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$ ./SRT
Nhap so tien trinh: 3
ProcessName, Process Arrival Time, Process Burst Time: 1 99 12
ProcessName, Process Arrival Time, Process Burst Time: 2 33 11
ProcessName, Process Arrival Time, Process Burst Time: 3 12 1
pName ArrTime BurstTime Start TAT Finish Waiting TimeLeft Res
3 12.0 1.0 12.0 1.0 13.0 0.0 0.0 0.0
2 33.0 11.0 33.0 11.0 44.0 0.0 0.0 0.0
1 99.0 12.0 99.0 12.0 111.0 0.0 0.0 0.0

Short Remaining Job First:
Average Waiting Time: 0
Average Turn Around Time: 8
pName ArrTime BurstTime Start TAT Finish Waiting TimeLeft Res
3 12.0 1.0 12.0 1.0 13.0 0.0 0.0 0.0
2 33.0 11.0 33.0 11.0 44.0 0.0 0.0 0.0
1 99.0 12.0 99.0 12.0 111.0 0.0 0.0 0.0
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$

```

- Các process đến trong các khoảng chênh lệch arrival time lớn hơn burst time nên các process được lập lịch tương tự như thuật toán FCFS

○ Test case 2:

```

phi-19522006@phi19522006-VirtualBox: ~/Documents/LAB 4
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$ ./SRT
Nhap so tien trinh: 5
ProcessName, Process Arrival Time, Process Burst Time: 1 0 8
ProcessName, Process Arrival Time, Process Burst Time: 2 2 19
ProcessName, Process Arrival Time, Process Burst Time: 3 4 3
ProcessName, Process Arrival Time, Process Burst Time: 4 5 6
ProcessName, Process Arrival Time, Process Burst Time: 5 7 10
pName  ArrTime  BurstTime  Start  TAT  Finish  Waiting  TimeLeft  Res
1      0.0      8.0        0.0    0.0   0.0     0.0      4.0      0.0
3      4.0      3.0        4.0    3.0   7.0     0.0      0.0      0.0
1      0.0      8.0        0.0   11.0  11.0     3.0      0.0      0.0
4      5.0      6.0       11.0  12.0  17.0     6.0      0.0      0.0
5      7.0     10.0       17.0  20.0  27.0    10.0      0.0      0.0
2      2.0     19.0       27.0  44.0  46.0    25.0      0.0      0.0

Short Remaining Job First:
Average Waiting Time: 8.8
Average Turn Around Time: 18
pName  ArrTime  BurstTime  Start  TAT  Finish  Waiting  TimeLeft  Res
1      0.0      8.0        0.0   11.0  11.0     3.0      0.0      0.0
3      4.0      3.0        4.0    3.0   7.0     0.0      0.0      0.0
4      5.0      6.0       11.0  12.0  17.0     6.0      0.0      0.0
5      7.0     10.0       17.0  20.0  27.0    10.0      0.0      0.0
2      2.0     19.0       27.0  44.0  46.0    25.0      0.0      0.0
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$

```

- Các tiến trình được thực thi lần lượt với thứ tự 1 3 1 4 5 2 như trên hình, phía trên dòng short remaining job first

○ Testcase 3:

```

phi-19522006@phi19522006-VirtualBox: ~/Documents/LAB 4
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$ ./SRT
Nhap so tien trinh: 5
ProcessName, Process Arrival Time, Process Burst Time: 1 0 12
ProcessName, Process Arrival Time, Process Burst Time: 2 2 7
ProcessName, Process Arrival Time, Process Burst Time: 3 5 8
ProcessName, Process Arrival Time, Process Burst Time: 4 9 3
ProcessName, Process Arrival Time, Process Burst Time: 5 12 6
pName  ArrTime  BurstTime  Start  TAT  Finish  Waiting  TimeLeft  Res
1      0.0     12.0        0.0    0.0   0.0     0.0     10.0      0.0
2      2.0      7.0        9.0    7.0   9.0     0.0      0.0      0.0
4      9.0      3.0        9.0    3.0  12.0     0.0      0.0      0.0
5     12.0      6.0       12.0    6.0  18.0     0.0      0.0      0.0
3      5.0      8.0       18.0   21.0  26.0    13.0      0.0      0.0
1      0.0     12.0        0.0   36.0  36.0    24.0      0.0      0.0

Short Remaining Job First:
Average Waiting Time: 7.4
Average Turn Around Time: 14.6
pName  ArrTime  BurstTime  Start  TAT  Finish  Waiting  TimeLeft  Res
1      0.0     12.0        0.0   36.0  36.0    24.0      0.0      0.0
2      2.0      7.0        9.0    7.0   9.0     0.0      0.0      0.0
4      9.0      3.0        9.0    3.0  12.0     0.0      0.0      0.0
5     12.0      6.0       12.0    6.0  18.0     0.0      0.0      0.0
3      5.0      8.0       18.0   21.0  26.0    13.0      0.0      0.0
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$

```

- Các tiến trình được thực thi lần lượt với thứ tự 1 2 4 5 3 1 như trên hình, phía trên dòng short remaining job first

3. Task 3: Viết chương trình mô phỏng giải thuật RR với các yêu cầu sau:

- ❖ Nhập số process
- ❖ Nhập quantum time
- ❖ Nhập process name, burst time
- ❖ In ra Gantt chart với các thông số: process name, start processor time, stop processor time
- ❖ In ra average waiting time và average turnaround time

- Code của giải thuật(Viết bằng C++)

```
- #include <iostream>
- #include <string>
- using namespace std;
-
- class Process
- {
- public:
-     int pname=0;
-     float paTime=0.f, pbTime=0.f;
-     float wTime = 0.f, taTime = 0.f;
-     float staTime = -1.f, fiTime=0.f;
-     float timeLeft = 0.f, resTime = 0.f;// use for RR and SRJF
-     //Function
-     void Nhap()
-     {
-         cout << "ProcessName, Process Arrival Time, Process Burst Time: ";
-         cin >> pname>> paTime>> pbTime;
-         timeLeft = pbTime;
-     }
-     void Xuat()
-     {
-         printf("%d      %8.1f\t%8.1f\t%8.1f\t%8.1f\t%8.1f\t%8.1f\t%8.1f",pname,paT
ime,pbTime,staTime, taTime,fiTime,wTime,timeLeft,resTime);
-         cout << "\n";
-     }
- };
- class Queue
- {
- public:
-     Process* p = NULL;
-     int soLuong = 0;
```

```

- Queue()
- {
-
- }
- ~Queue()
- {
-     if(soLuong!=0)
-         delete p;
- }
- void Push(Process temp)
- {
-     Process* t = new Process[soLuong];
-     for (int i = 0; i < soLuong; i++)
-     {
-         t[i] = p[i];
-     }
-     soLuong += 1;
-     p = new Process[soLuong];
-     for (int i = 0; i < soLuong-1; i++)
-     {
-         p[i] = t[i];
-     }
-     p[soLuong - 1] = temp;
-     delete t;
- }
- Process Pop()
- {
-     if (soLuong == 0)
-     {
-         Process error;
-         error.pname = -1;
-         return error;
-     }
-     else
-     {
-         Process res = p[0];
-         for (int i = 1; i < soLuong; i++)
-         {
-             p[i-1] = p[i];

```

```

-         }
-         soLuong -= 1;
-         return res;
-     }
- }
- Process PopAtMin()
- {
-     if (soLuong == 0)
-     {
-         Process error;
-         error.pname = -1;
-         return error;
-     }
-     else
-     {
-         float bTimeMin = p[0].pbTime;
-         for (int i = 1; i < soLuong; i++)
-         {
-             if (p[i].pbTime < bTimeMin)
-             {
-                 bTimeMin = p[i].pbTime;
-             }
-         }
-         Process res;
-         int index=0;
-         for (int i = 0; i < soLuong; i++)
-         {
-             if (p[i].pbTime == bTimeMin)
-             {
-                 res = p[i];
-                 index = i;
-                 break;
-             }
-         }
-         Process* t = new Process[soLuong - 1];
-         for (int i = 0; i < index; i++)
-         {
-             t[i] = p[i];
-         }
-     }
- }

```



```

-         for (int i = index + 1 ; i < soLuong; i++)
-         {
-             t[i-1] = p[i];
-         }
-         soLuong -= 1;
-         p= new Process[soLuong];
-         for (int i = 0; i < soLuong; i++)
-         {
-             p[i] = t[i];
-         }
-         delete t;
-         return res;
-     }
- }
- Process ViewAtMin()
- {
-     float TimeMin = p[0].timeLeft;
-     for (int i = 1; i < soLuong; i++)
-     {
-         if (p[i].timeLeft < TimeMin)
-         {
-             TimeMin = p[i].pbTime;
-         }
-     }
-     Process res;
-     int index = 0;
-     for (int i = 0; i < soLuong; i++)
-     {
-         if (p[i].timeLeft == TimeMin)
-         {
-             res = p[i];
-             index = i;
-             break;
-         }
-     }
-     return res;
- }
- Process PopAtMin2()
- {

```

```

-     if (soLuong == 0)
-     {
-         Process error;
-         error.pname = -1;
-         return error;
-     }
-     else
-     {
-         float TimeMin = p[0].timeLeft;
-         for (int i = 1; i < soLuong; i++)
-         {
-             if (p[i].timeLeft < TimeMin)
-             {
-                 TimeMin = p[i].timeLeft;
-             }
-         }
-         Process res;
-         int index = 0;
-         for (int i = 0; i < soLuong; i++)
-         {
-             if (p[i].timeLeft == TimeMin)
-             {
-                 res = p[i];
-                 index = i;
-                 break;
-             }
-         }
-         Process* t = new Process[soLuong - 1];
-         for (int i = 0; i < index; i++)
-         {
-             t[i] = p[i];
-         }
-         for (int i = index + 1; i < soLuong; i++)
-         {
-             t[i - 1] = p[i];
-         }
-         soLuong -= 1;
-         p = new Process[soLuong];
-         for (int i = 0; i < soLuong; i++)

```

```

-         {
-             p[i] = t[i];
-         }
-         delete t;
-         return res;
-     }
- }
- bool isEmpty()
- {
-     if (soLuong == 0) return true;
-     else return false;
- }
- void ViewQueue()
- {
-     for (int i = 0; i < soLuong; i++)
-     {
-         printf("%d      %8.1f\t%8.1f\t%8.1f", p[i].pname, p[i].paTime, p[i].pbTime, p
[i].staTime);
-         cout << "\n";
-     }
- }
- Process PopAndMerge()
- {
-     Process first = p[0];
-     int indexoffirst=0;
-     for (int i = 1; i < soLuong; i++)
-     {
-         if (first.pname == p[i].pname)
-         {
-             first=p[i];
-             for (int j = indexoffirst; j < soLuong - 1; j++)
-             {
-                 p[j] = p[j + 1];
-             }
-             indexoffirst = i - 1;
-             soLuong -= 1;
-             /*cout << "Bop lan 1: \n";
-             ViewQueue();

```

```

-         cout << "\n";*/
-     }
- }
-     for (int j = indexoffirst; j < soLuong - 1; j++)
-     {
-         p[j] = p[j + 1];
-     }
-     soLuong -= 1;
-     /*cout << "Queue ne: \n";
-     this->ViewQueue();
-     cout << "\n";*/
-     return first;
- }
- };

```

- Các class Process và Queue chứa các hàm và thuộc tính giống như hai bài ở trên

```

class Algorithm
{
public:
    Process pc[100];
    int soLuong=0;
    float avgWTime = 0.f, avgTATime = 0.f;
    void Nhap()
    {
        cout << "Nhap so tien trinh: ";
        int n; cin >> n;
        soLuong = n;
        for (int i = 0; i < n; i++)
        {
            pc[i].Nhap();
        }
    }
    void Xuat()
    {
        cout << "pName    ArrTime    BurstTime        Start        TAT        Finish
Waiting    TimeLeft    Res\n";
        for (int i = 0; i < soLuong; i++)
        {
            pc[i].Xuat();
        }
    }
    void RoundRobin()

```

```

{
    Queue pQueue;

    //Process* Ketqua = new Process[soLuong];
    Queue run;
    float RRtime=0.f;
    //Đóng cho c❖i chart
    Queue chart;
    float Time[100];
    int m = 0;
    //S?p x?p c❖c ti?n tr?nh theo arrival time
    for (int i = 0; i < soLuong - 1; i++)
    {
        for (int j = i + 1; j < soLuong; j++)
        {
            if (pc[i].paTime > pc[j].paTime)
            {
                Process temp = pc[i];
                pc[i] = pc[j];
                pc[j] = temp;
            }
        }
    }
    cout << "Nhap RR time: ";
    cin >> RRtime;
    while (RRtime <= 0)
    {
        cout << "Nhap lai RR time: ";
        cin >> RRtime;
    }
    cout << "pName    ArrTime    BurstTime    Start    TAT    Finish\n";
    cout << "Waiting    TimeLeft    Res\n";
    float time = pc[0].paTime;
    int index = 0;

    //lấy các process ở thời điểm đầu tiên
    while (index < soLuong && pc[index].paTime <= time)
    {
        pQueue.Push(pc[index]);
        index++;
    }
    Process current;
    while (true)
    {
        if (pQueue.isEmpty() == true)
            break;

```

```

current = pQueue.Pop();
if (current.timeLeft > RRtime)
{
    //C?p nh?t th?ng s? c?a project
    if (current.staTime == -1)
    {
        current.staTime = time;
    }
    current.timeLeft -= RRtime;
    //D?ng cho c?i chart
    Time[m] = time;
    m++;

    time += RRtime;
    while (index < soLuong && pc[index].paTime <= time)
    {
        pQueue.Push(pc[index]);
        index++;
    }
    pQueue.Push(current);
    current.Xuat();
}
else if (current.timeLeft == RRtime)
{
    //C?p nh?t th?ng s? c?a process
    if (current.staTime == -1)
    {
        current.staTime = time;
    }
    current.timeLeft -= RRtime;
    current.fiTime = time + RRtime;
    current.taTime = current.fiTime - current.paTime;
    current.wTime = current.fiTime - current.paTime - current.pbTime;
    current.resTime = current.staTime - current.paTime;
    //D?ng cho c?i chart
    Time[m] = time;
    m++;

    time += RRtime;
    while (index < soLuong && pc[index].paTime <= time)
    {
        pQueue.Push(pc[index]);
        index++;
    }
    run.Push(current);
    current.Xuat();
}

```

```

    }
    else if (current.timeLeft < RRtime)
    {
        if (current.staTime == -1)
        {
            current.staTime = time;
        }
        current.fiTime = time + current.timeLeft;
        current.taTime = current.fiTime - current.paTime;
        current.wTime = current.fiTime - current.paTime - current.pbTime;
        current.resTime = current.staTime - current.paTime;
        //Đóng cho còi chart
        Time[m] = time;
        m++;

        time += current.timeLeft;
        current.timeLeft = 0;
        while (index < soLuong && pc[index].paTime <= time)
        {
            pQueue.Push(pc[index]);
            index++;
        }
        run.Push(current);
        current.Xuat();

        if (pQueue.isEmpty() == true)
        {
            time = pc[index].paTime;
            while (index < soLuong && pc[index].paTime <= time)
            {
                pQueue.Push(pc[index]);
                index++;
            }
        }
        chart.Push(current);
    }
    int i = 0;
    while (run.isEmpty() != true)
    {
        pc[i] = run.Pop();
        i++;
    }
    //Đóng cho còi chart
    Time[m] = time;
    m++;

```

```

//Tính avgWaiting và avgTAT
for (int i = 0; i < soLuong; i++)
{
    avgWTime += pc[i].wTime;
    avgTATime += pc[i].taTime;
    pc[i].resTime = pc[i].staTime - pc[i].paTime;
}
avgWTime /= soLuong;
avgTATime /= soLuong;
cout << "\nRound Robin:\nAverage Waiting Time: " << avgWTime << "\nAverage Turn Around Time: " << avgTATime << "\n";

cout << "\nGantt chart:\n";
Process k;
while (chart.isEmpty() != true)
{
    k = chart.Pop();
    printf("|-- %d --", k.pname);
    i++;
}
cout << "|\n";
for (int i = 0; i < m; i++)
{
    if (Time[i] >= 100)
    {
        cout << Time[i] << " ";
    }
    else if (Time[i] < 10)
    {
        cout << Time[i] << " ";
    }
    else
        cout << Time[i] << " ";
}
cout << "\n\n";
}
};

```

- Class Algorithm giúp ta thực hiện các thao tác nhập xuất nhiều tiến trình và in kết quả ra bên ngoài màn hình và có hàm Round Robin giúp ta lập lịch các tiến trình đã nhập theo thuật toán Round Robin:
- Các bước thực hiện của hàm Round Robin:
 - o B1: Thực hiện sắp xếp các tiến trình theo arrival time thấp nhất đến arrival time cao nhất
 - o B2: Chọn arrival time của tiến trình đầu tiên là thời gian hiện tại

- B3: Thêm tất cả các tiến trình có arrival time bằng với thời gian hiện tại push vào trong queue, Pop và thực thi tiến trình đầu tiên trong Queue với khoảng thời gian là quantum time hoặc sẽ thực hiện cho đến kết thúc nếu quantum time lớn hơn time left.
- B4: Tiếp tục thực hiện B3 cho đến khi hết tiến trình hoặc hàng đợi trống thì lấy thời gian tiếp theo bằng thời gian của tiến trình tiếp theo nữa đến(trường hợp còn tiến trình).
- B5: Kết thúc và in kết quả ra ngoài màn hình

```

- int main()
- {
-     Algorithm a;
-     a.Nhap();
-     a.RoundRobin();
-     a.Xuat();
- }

```

- Chương trình chính giúp ta nhập các process, thực hiện lập lịch các tiến trình theo giải thuật Round Robin và xuất ra ngoài màn hình
- Chạy trên Ubuntu:
 - Testcase 1:

```

phi-19522006@phi19522006-VirtualBox: ~/Documents/LAB 4
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$ g++ RR.cpp -o RR
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$ ./RR
Nhap so tien trinh: 3
ProcessName, Process Arrival Time, Process Burst Time: 1 99 12
ProcessName, Process Arrival Time, Process Burst Time: 2 33 11
ProcessName, Process Arrival Time, Process Burst Time: 3 12 1
Nhap RR time: 4
pName  ArrTime  BurstTime  Start  TAT  Finish  Waiting  Res
3      12.0    1.0       12.0   1.0   13.0    0.0      0.0
2      33.0    11.0      33.0   0.0   0.0     0.0      0.0
2      33.0    11.0      33.0   0.0   0.0     0.0      0.0
2      33.0    11.0      33.0   11.0  44.0    0.0      0.0
1      99.0    12.0      99.0   0.0   0.0     0.0      0.0
1      99.0    12.0      99.0   0.0   0.0     0.0      0.0
1      99.0    12.0      99.0   12.0  111.0   0.0      0.0

Round Ronin:
Average Waiting Time: 0
Average Turn Around Time: 8

Gantt chart:
|-- 3  --|-- 2  --|-- 2  --|-- 2  --|-- 1  --|-- 1  --|-- 1  --|
12      33      37      41      99      103      107      111

pName  ArrTime  BurstTime  Start  TAT  Finish  Waiting  Res
3      12.0    1.0       12.0   1.0   13.0    0.0      0.0
2      33.0    11.0      33.0   11.0  44.0    0.0      0.0
1      99.0    12.0      99.0   12.0  111.0   0.0      0.0
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$

```

- - Các tiến trình với thời gian arrival time chênh nhau khá lớn và burst time lớn hơn quantum time nên sẽ bị thực hiện từng phần theo quantum time.

○ Testcase 2:

```
phi-19522006@phi19522006-VirtualBox: ~/Documents/LAB 4 $ ./RR
Nhap so tien trinh: 5
ProcessName, Process Arrival Time, Process Burst Time: 1 0 12
ProcessName, Process Arrival Time, Process Burst Time: 2 2 7
ProcessName, Process Arrival Time, Process Burst Time: 3 5 8
ProcessName, Process Arrival Time, Process Burst Time: 4 9 3
ProcessName, Process Arrival Time, Process Burst Time: 5 12 6
Nhap RR time: 4
pName  ArrTime  BurstTime  Start  TAT  Finish  Waiting  Res
1      0.0      12.0      0.0    0.0    0.0      0.0      0.0
2      2.0      7.0       4.0    0.0    0.0      0.0      0.0
1      0.0      12.0      0.0    0.0    0.0      0.0      0.0
3      5.0      8.0     12.0    0.0    0.0      0.0      0.0
2      2.0      7.0       4.0   17.0   19.0     10.0      2.0
4      9.0      3.0     19.0   13.0   22.0     10.0     10.0
5     12.0      6.0     22.0    0.0    0.0      0.0      0.0
1      0.0      12.0      0.0   30.0   30.0     18.0      0.0
3      5.0      8.0     12.0   29.0   34.0     21.0      7.0
5     12.0      6.0     22.0   24.0   36.0     18.0     10.0

Round Ronin:
Average Waiting Time: 15.4
Average Turn Around Time: 22.6

Gantt chart:
|-- 1  --|-- 2  --|-- 1  --|-- 3  --|-- 2  --|-- 4  --|-- 5  --|-- 1  --|-- 3  --|-- 5  --|
0      4      8      12      16      19      22      26      30      34      36

pName  ArrTime  BurstTime  Start  TAT  Finish  Waiting  Res
2      2.0      7.0       4.0   17.0   19.0     10.0      2.0
4      9.0      3.0     19.0   13.0   22.0     10.0     10.0
1      0.0      12.0      0.0   30.0   30.0     18.0      0.0
3      5.0      8.0     12.0   29.0   34.0     21.0      7.0
5     12.0      6.0     22.0   24.0   36.0     18.0     10.0
```

- Quantum time = 4 và các tiến trình có arrival time không chênh nhau quá lớn nên các tiến trình được thực hiện thay phiên nhau

○ Testcase 3:

```
phi-19522006@phi19522006-VirtualBox: ~/Documents/LAB 4 $ ./RR
Nhap so tien trinh: 5
ProcessName, Process Arrival Time, Process Burst Time: 1 0 8
ProcessName, Process Arrival Time, Process Burst Time: 2 2 19
ProcessName, Process Arrival Time, Process Burst Time: 3 4 3
ProcessName, Process Arrival Time, Process Burst Time: 4 5 6
ProcessName, Process Arrival Time, Process Burst Time: 5 7 10
Nhap RR time: 22
pName  ArrTime  BurstTime  Start  TAT  Finish  Waiting  Res
1      0.0      8.0       0.0    8.0    8.0      0.0      0.0
2      2.0     19.0       8.0   25.0   27.0      6.0      6.0
3      4.0      3.0     27.0   26.0   30.0     23.0     23.0
4      5.0      6.0     30.0   31.0   36.0     25.0     25.0
5      7.0     10.0     36.0   39.0   46.0     29.0     29.0

Round Ronin:
Average Waiting Time: 16.6
Average Turn Around Time: 25.8

Gantt chart:
|-- 1  --|-- 2  --|-- 3  --|-- 4  --|-- 5  --|
0      8     27     30     36     46

pName  ArrTime  BurstTime  Start  TAT  Finish  Waiting  Res
1      0.0      8.0       0.0    8.0    8.0      0.0      0.0
2      2.0     19.0       8.0   25.0   27.0      6.0      6.0
3      4.0      3.0     27.0   26.0   30.0     23.0     23.0
4      5.0      6.0     30.0   31.0   36.0     25.0     25.0
5      7.0     10.0     36.0   39.0   46.0     29.0     29.0
```

- Trường hợp này ta có quantum time quá lớn nên các tiến trình giống như được lập lịch theo thuật toán FCFS.
- Testcase 4 là 5 là yêu cầu của đề bài khi arrival time của tất cả tiến trình đều bằng 0

```

phi-19522006@phi19522006-VirtualBox: ~/Documents/LAB 4
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$ ./RR
Nhap so tien trinh: 3
ProcessName, Process Arrival Time, Process Burst Time: 1 0 3
ProcessName, Process Arrival Time, Process Burst Time: 2 0 4
ProcessName, Process Arrival Time, Process Burst Time: 5 0 1
Nhap RR time: 2
pName  ArrTime  BurstTime  Start    TAT    Finish    Waiting    Res
1      0.0      3.0        0.0      0.0    0.0        0.0        0.0
2      0.0      4.0        2.0      2.0    2.0        0.0        0.0
5      0.0      1.0        4.0      5.0    5.0        4.0        4.0
1      0.0      3.0        0.0      6.0    6.0        3.0        0.0
2      0.0      4.0        2.0      8.0    8.0        4.0        2.0

Round Ronin:
Average Waiting Time: 3.66667
Average Turn Around Time: 6.33333

Gantt chart:
|-- 1 --|-- 2 --|-- 5 --|-- 1 --|-- 2 --|
0      2      4      5      6      8

pName  ArrTime  BurstTime  Start    TAT    Finish    Waiting    Res
5      0.0      1.0        4.0      5.0    5.0        4.0        4.0
1      0.0      3.0        0.0      6.0    6.0        3.0        0.0
2      0.0      4.0        2.0      8.0    8.0        4.0        2.0
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$

```

- Các tiến trình được thực hiện ngẫu nhiên thay phiên nhau

```

phi-19522006@phi19522006-VirtualBox: ~/Documents/LAB 4
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$ ./RR
Nhap so tien trinh: 3
ProcessName, Process Arrival Time, Process Burst Time: 1 0 3
ProcessName, Process Arrival Time, Process Burst Time: 2 0 4
ProcessName, Process Arrival Time, Process Burst Time: 3 0 4
Nhap RR time: 1
pName  ArrTime  BurstTime  Start    TAT    Finish    Waiting    Res
1      0.0      3.0        0.0      0.0    0.0        0.0        0.0
2      0.0      1.0        1.0      2.0    2.0        1.0        1.0
3      0.0      4.0        2.0      0.0    0.0        0.0        0.0
1      0.0      3.0        0.0      0.0    0.0        0.0        0.0
3      0.0      4.0        2.0      0.0    0.0        0.0        0.0
1      0.0      3.0        0.0      6.0    6.0        3.0        0.0
3      0.0      4.0        2.0      0.0    0.0        0.0        0.0
3      0.0      4.0        2.0      8.0    8.0        4.0        2.0

Round Ronin:
Average Waiting Time: 2.66667
Average Turn Around Time: 5.33333

Gantt chart:
|-- 1 --|-- 2 --|-- 3 --|-- 1 --|-- 3 --|-- 1 --|-- 3 --|--
0      1      2      3      4      5      6      7      8

pName  ArrTime  BurstTime  Start    TAT    Finish    Waiting    Res
2      0.0      1.0        1.0      2.0    2.0        1.0        1.0
1      0.0      3.0        0.0      6.0    6.0        3.0        0.0
3      0.0      4.0        2.0      8.0    8.0        4.0        2.0
phi-19522006@phi19522006-VirtualBox:~/Documents/LAB 4$

```

- Khi chọn quantum time quá nhỏ thì các tiến trình thực hiện thay phiên nhau liên tục và khá lãng phí chi phí chuyển ngữ cảnh