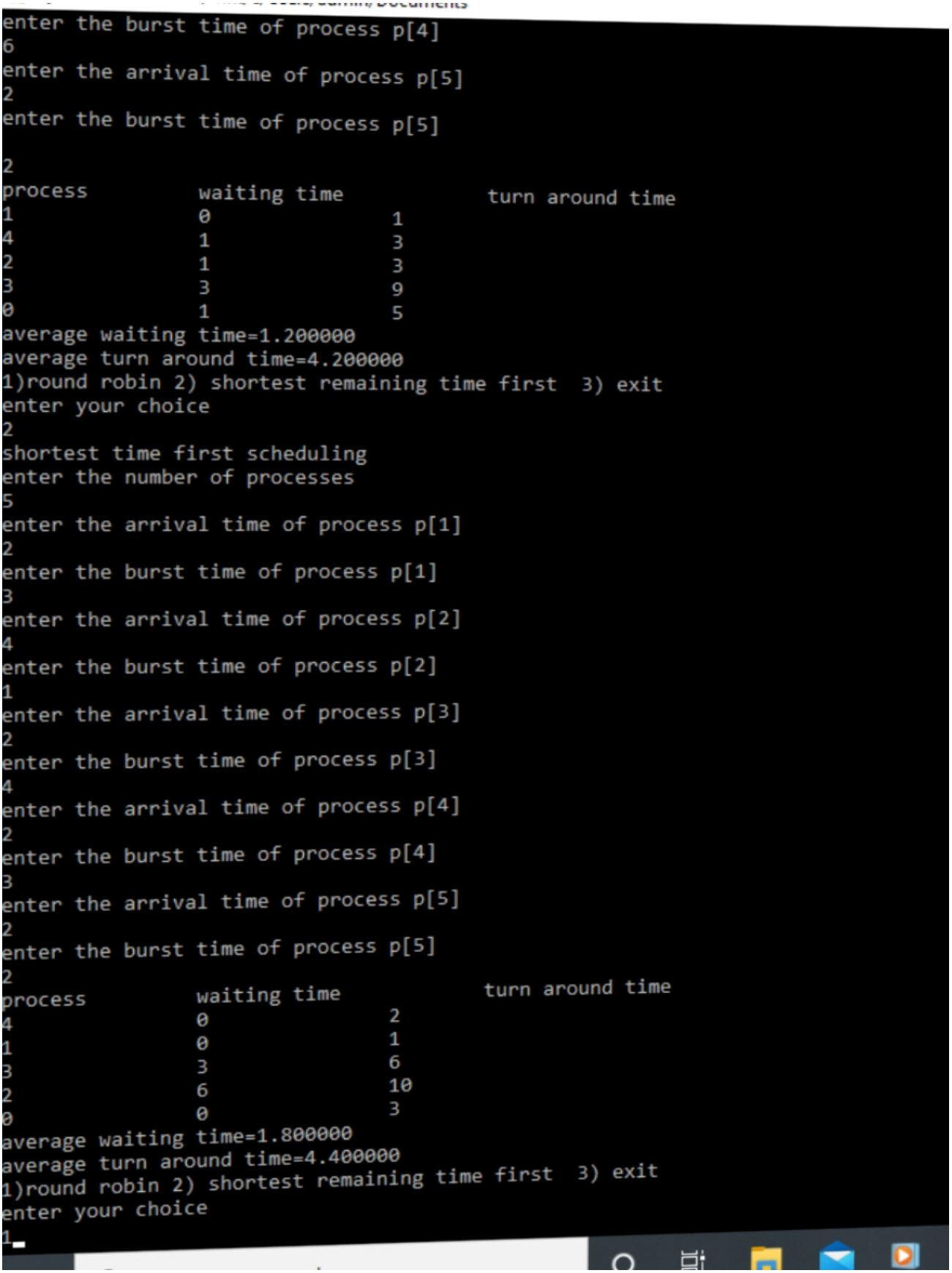
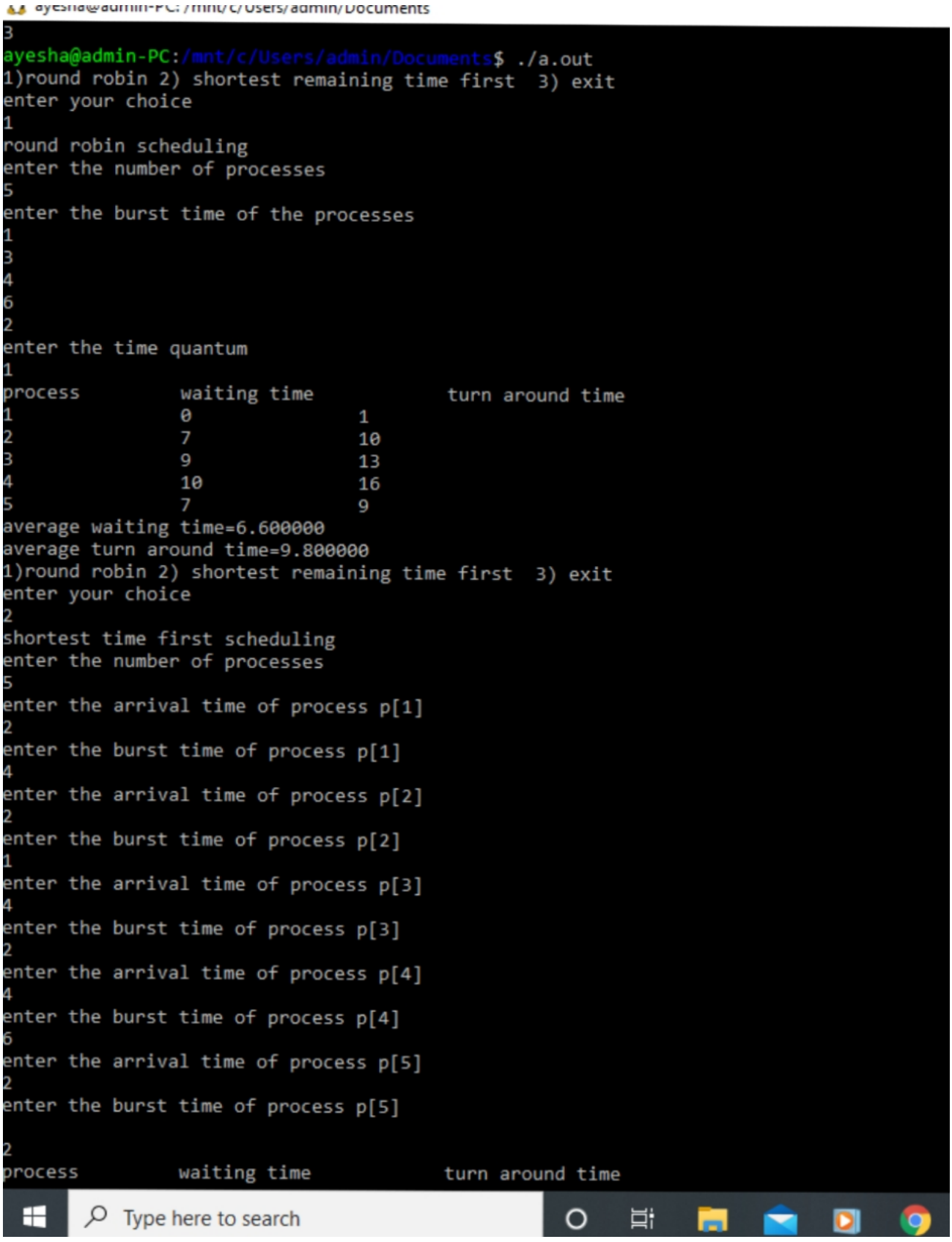
ayesha@admin-PC: /mnt/c/Users/admin/Documents

```
2
enter the burst time of process p[1]
3
enter the arrival time of process p[2]
4
enter the burst time of process p[2]
1
enter the arrival time of process p[3]
2
enter the burst time of process p[3]
4
enter the arrival time of process p[4]
2
enter the burst time of process p[4]
3
enter the arrival time of process p[5]
2
enter the burst time of process p[5]
2
process         waiting time            turn around time
4               0               2
1               0               1
3               3               6
2               6               10
0               0               3
average waiting time=1.800000
average turn around time=4.400000
1)round robin 2) shortest remaining time first  3) exit
enter your choice
1
round robin scheduling
enter the number of processes
5
enter the burst time of the processes
2
4
2
4
1
enter the time quantum
2
process         waiting time            turn around time
1               0               2
2               7               11
3               4               6
4               9               13
5               8               9
average waiting time=5.600000
average turn around time=8.200000
1)round robin 2) shortest remaining time first  3) exit
enter your choice
```


enter the burst time of process p[4]
6
enter the arrival time of process p[5]
2
enter the burst time of process p[5]
2
process waiting time turn around time
1 0 1
4 1 3
2 1 3
3 3 9
0 1 5
average waiting time=1.200000
average turn around time=4.200000
1)round robin 2) shortest remaining time first 3) exit
enter your choice
2
shortest time first scheduling
enter the number of processes
5
enter the arrival time of process p[1]
2
enter the burst time of process p[1]
3
enter the arrival time of process p[2]
4
enter the burst time of process p[2]
1
enter the arrival time of process p[3]
2
enter the burst time of process p[3]
4
enter the arrival time of process p[4]
2
enter the burst time of process p[4]
3
enter the arrival time of process p[5]
2
enter the burst time of process p[5]
2
process waiting time turn around time
4 0 2
1 0 1
3 3 6
2 6 10
0 0 3
average waiting time=1.800000
average turn around time=4.400000
1)round robin 2) shortest remaining time first 3) exit
enter your choice
1


ayesha@admin-PC: /mnt/c/Users/admin/Documents
3
ayesha@admin-PC:/mnt/c/Users/admin/Documents$ ./a.out
1)round robin 2) shortest remaining time first 3) exit
enter your choice
1
round robin scheduling
enter the number of processes
5
enter the burst time of the processes
1
3
4
6
2
enter the time quantum
1
process waiting time turn around time
1 0 1
2 7 10
3 9 13
4 10 16
5 7 9
average waiting time=6.600000
average turn around time=9.800000
1)round robin 2) shortest remaining time first 3) exit
enter your choice
2
shortest time first scheduling
enter the number of processes
5
enter the arrival time of process p[1]
2
enter the burst time of process p[1]
4
enter the arrival time of process p[2]
2
enter the burst time of process p[2]
1
enter the arrival time of process p[3]
4
enter the burst time of process p[3]
2
enter the arrival time of process p[4]
4
enter the burst time of process p[4]
6
enter the arrival time of process p[5]
2
enter the burst time of process p[5]
2
process waiting time turn around time
Type here to search

Scheduling queues

Job queue:

Job queue contains the set of all the processes in the system. As processes enter the system they are put into a job queue.

Ready queue:

Ready queue contains the set of all processes residing in the main memory awaiting execution.

The aim of scheduling is to make the system efficient, ~~safe~~ fast and fair.

Whenever the CPU becomes idle the operating system selects one of the processes from the ready queue.

The objective of multiprogramming is to have some process running at all times to maximize CPU utilization.

FCFS

First Come First Serve is an operating system scheduling algorithm that automatically executes queued requests

and processes in of their arrival. It is the easiest and simplest CPU scheduling algorithm.

Advantages

It is the simplest algorithm

It is easy to program as it is based on First come first serve

Disadvantages

It is non preemptive and once a process is allocated it is completed.

The average waiting time is high.

Shortest Job Scheduling

Assumes I we know the length of the next CPU burst of all ready processes

SJF estimates the length of the next burst based on the lengths of recent CPU bursts.

Two types:

Preemptive

Non preemptive

Advantages

SJF scheduling reduces the average waiting time compared to FCFS.

It is appropriate when jobs are run in batches.

Drawbacks

The algorithm can cause very long turn around times or starvation.

It is not practical to implement

Round Robin Scheduling

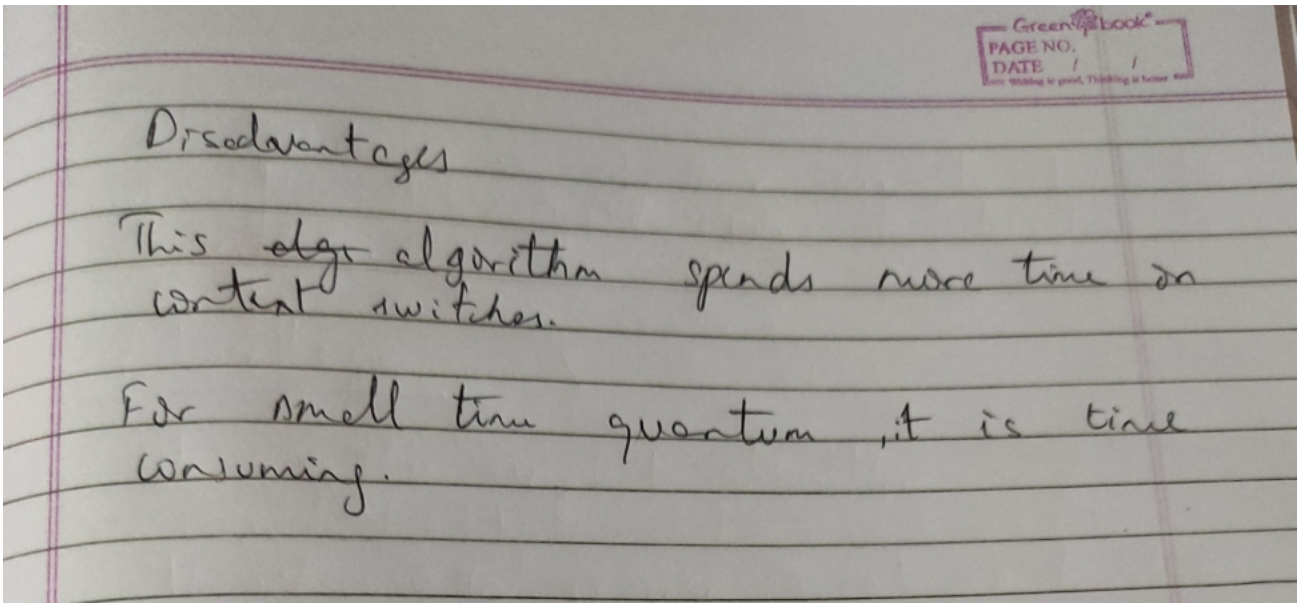
It uses the Round Robin principle where each person gets an equal share of something in turns.

It is the oldest, simplest scheduling algorithm which is mostly used for multitasking.

Advantages

Doesn't face starvation

All jobs get fair allocation of CPU

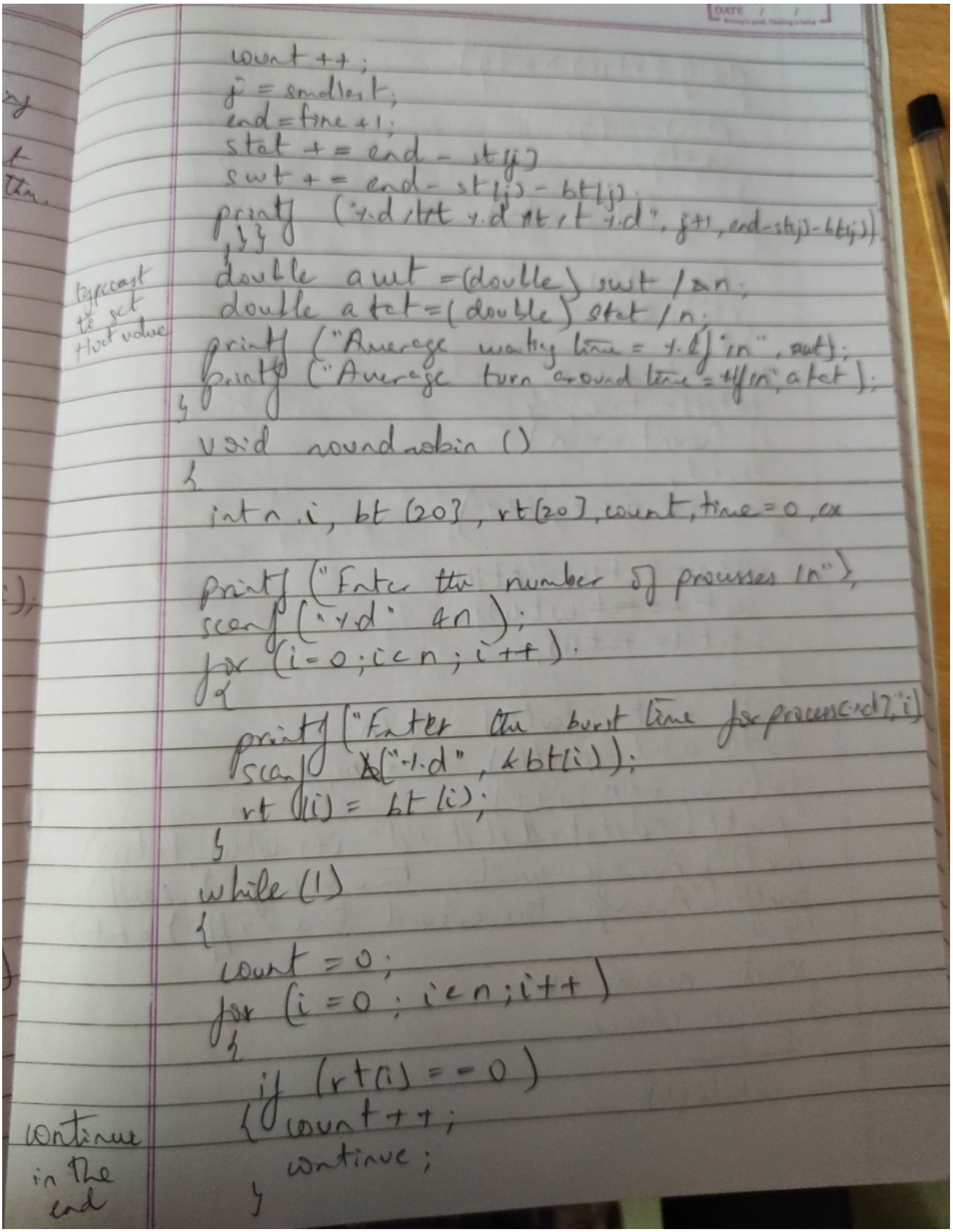

Green book
PAGE NO.
DATE / /
Disadvantages
This algorithm spends more time on context switches.
For small time quantum, it is time consuming.

7) Design, develop and implement a c/c++ or Java program to simulate the working of shortest remaining time and round robin scheduling algorithms. Experiment with different quantum sizes for the algorithm.

```
#include<stdio.h>
#include<stdlib.h>
void srtf()
{ int n,i,bt[20], rt[20], st[20], time, count=0;
  int smallest, j, stct=0, swt=0
  rt[20]=999;
  printf("Enter the number of processes \n");
  scanf("%d", &n);
  prin
  for(i=0; i<n; i++)
{ printf("Enter the arrival time for process[%d]", i);
  scanf("%d", &st[i]);
  printf("Enter the burst time for process [%d]", i);
  scanf("%d", &bt[i]);
  rt[i] = bt[i];
}
for(time=0; count!=n; time++)
{ smallest=20;
  for(i=0; i<n; i++)
{ smallest=20;
  if((rt[i] < rt[smallest]) && (rt[i]>0) && st[i]<=time)
  smallest=i;
}rt[smallest]--;
 if(rt[smallest]==0)
 {
```

required

typecast to float



```
        count++;
        j = smallest;
        end = time + 1;
        stat += end - st[j];
        swt += end - st[j] - bt[j];
        printf("%d\t\t%d\t\t%d", j+1, end-st[j]-bt[j]);
    }
    double awt = (double) swt / n;
    double atat = (double) stat / n;
    printf("Average waiting time = %.6f\n", awt);
    printf("Average turn around time = %f\n", atat);
}

void roundrobin()
{
    int n, i, bt[20], rt[20], count, time=0, q
    printf("Enter the number of processes\n");
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
        printf("Enter the burst time for process %d", i);
        scanf("%d", &bt[i]);
        rt[i] = bt[i];
    }
    while(1)
    {
        count = 0;
        for(i=0; i<n; i++)
        {
            if(rt[i] == 0)
            {
                count++;
                continue;
            }
```

typecast to get float value

continue in the end

```
else if (rt[i] >= tq)
{
  rt[i] -= tq;
  ex = tq;
}
else if (rt[i] < tq)
{
  ex = rt[i];
  rt[i] = 0;
}
time = time + ex;
tat[i] = time;
if (count == n)
  break;
for(i=0; i<n; i++)
{
  wt[i] = tat[i] - bt[i];
  swt += wt[i];
  stat += tat[i];
}
double awt = (double) swt/n;
double atat = (double) stat/n;
for(i=0; i<n; i++)
  printf("%d\t\t%d\t\t%d", i+1, wt[i], tat[i]);
printf("Average waiting time = %lf", awt);
printf("Average turn around time = %lf", atat);
}
void main()
{
  int choice;
  system
  while(1)
```

```
int
{
printf (" 1) Round Robin 2) SRTF 3) Exit \n");
printf(" Enter your choice \n");
scanf (" %d", &choice);
switch (choice)
{
case 1:
roundrobin()
break;
case 2:
srtf ();
break;
case 3:
exit(0);
}
}
}
```

PDF Created Using

Camera Scanner

Easily Scan documents & Generate PDF

https://play.google.com/store/apps/details?id=photo.pdf.maker