OS_Lab2

K. Ram Mohan

COE19B055

Q1) FCFS

```
#include<stdio.h>
#include<stdlib.h>
#define SIZE 100

void swap(int *p,int *q)
{
   int temp;
   temp=*p;
   *p=*q;
   *q=temp;
}

//function to sort based on arrival time
void sort_arr(int id[], int arr[], int burst_time[] , int size)
{
   int i, j;
   int go=1;
   while(go)
   {
      for(i=0; i<size-1; i++)
      {
         go=0;
         for(j=0; j<size-1-i; j++)
         {
            if(arr[j]>arr[j+1])
```

```
            {
                swap(&arr[j], &arr[j+1]);
                swap(&id[j], &id[j+1]);
                swap(&burst_time[j], &burst_time[j+1]);
                go=1;
            }
        }
    }
  }
}

int main()
{
  int total_process, i;
  int process_id[SIZE], arr_time[SIZE], burst_time[SIZE], time=0;
  float tot_waiting_time=0, tot_trt=0;

  printf("Enter no of process: ");
  scanf("%d", &total_process);

  for(i=0; i<total_process; i++)
  {
    printf("Process_id of process %d: ", i+1);
    scanf("%d", &process_id[i]);
    printf("Arrival time of process %d: ", i+1);
    scanf("%d", &arr_time[i]);
    printf("Burst time of process %d: ", i+1);
    scanf("%d", &burst_time[i]);
  }
```

```
sort_arr(process_id, arr_time, burst_time, total_process);
int completion_time[SIZE], waiting_time[SIZE], turn_arnd_time[SIZE];

for(i=0; i<total_process; i++)
{
   //for initial one and for cpu idle time one
   if(time==0 || time<arr_time[i])
   {
      completion_time[i] = arr_time[i] + burst_time[i];
   }
   else
   {
      completion_time[i] = completion_time[i-1] + burst_time[i];
   }
   time = completion_time[i];
}

for(i=0; i<total_process; i++)
{
   turn_arnd_time[i] = completion_time[i] - arr_time[i];
   tot_trt = tot_trt + turn_arnd_time[i];
}

for(i=0; i<total_process; i++)
{
   waiting_time[i] = turn_arnd_time[i] - burst_time[i];
   tot_waiting_time = tot_waiting_time + waiting_time[i];
}
printf("Processes arrival_time Burst time completion time Turn around time Waiting time\n");
```

```
    for(i=0; i<total_process; i++)
    {
        printf("%d", process_id[i]);
        printf("\t \t %d", arr_time[i]);
        printf("\t \t %d", burst_time[i]);
        printf("\t \t %d", completion_time[i]);
        printf("\t \t %d", turn_arnd_time[i]);
        printf("\t \t %d \n", waiting_time[i]);
    }
    float avg_wait = tot_waiting_time/total_process;
    float avg_trt = tot_trt/total_process;
    printf("Average waiting time is : %f\n", avg_wait);
    printf("Average turn around time is: %f", avg_trt);
}
```

a) All Input with arrival time “0”

COF 19BOSS
K.Ram Mohan

Q1) a) Arrival time = 0 FCFS

no:of process = 6

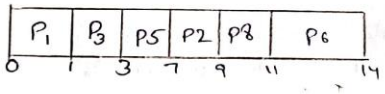
Input:

PID	AT	BT
1	0	1
3	0	2
5	0	4
2	0	2
8	0	2
6	0	3

OUTPUT:

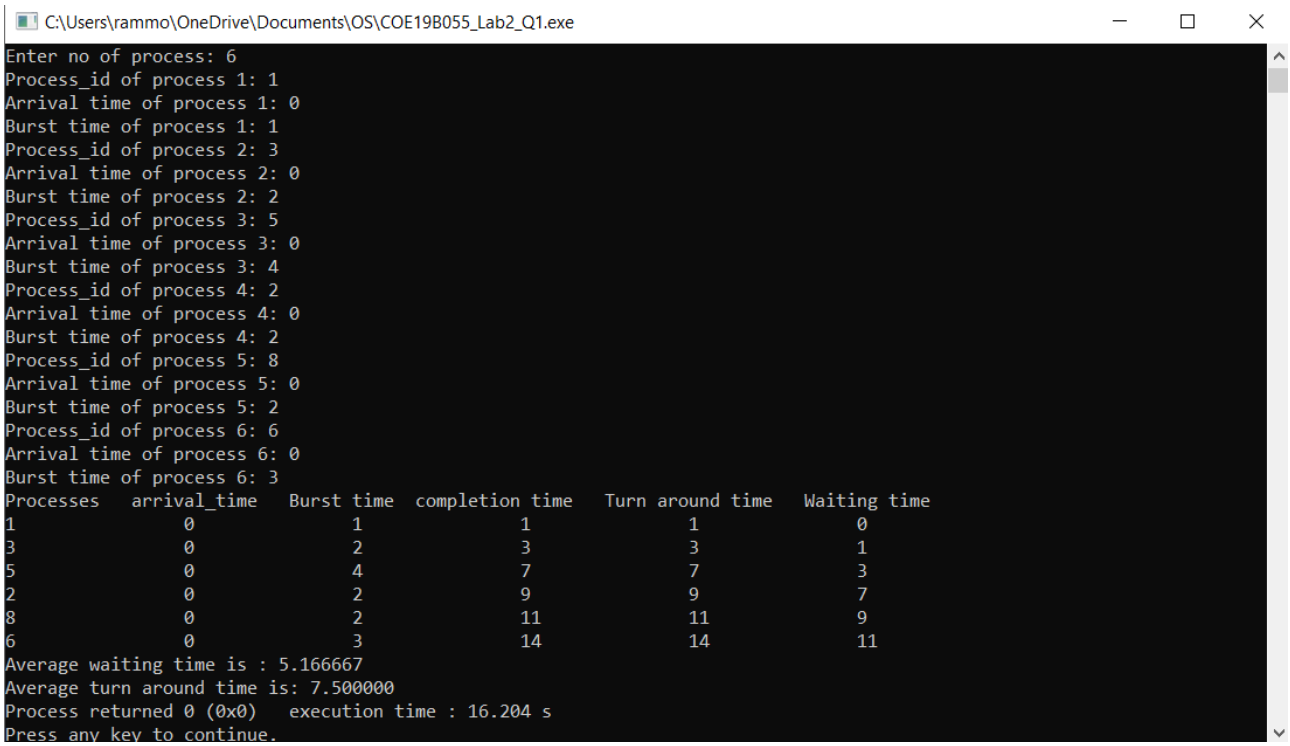
PID	AT	BT	CT	TAT	WT
1	0	1	1	1	0
3	0	2	3	3	1
5	0	4	7	7	3
2	0	2	9	9	7
8	0	2	11	11	9
6	0	3	14	14	11

GANTT CHART:



Average waiting time = $\frac{31}{6}$

Average turn around time = $\frac{45}{6}$


C:\Users\rammo\OneDrive\Documents\OS\COE19B055_Lab2_Q1.exe
Enter no of process: 6
Process_id of process 1: 1
Arrival time of process 1: 0
Burst time of process 1: 1
Process_id of process 2: 3
Arrival time of process 2: 0
Burst time of process 2: 2
Process_id of process 3: 5
Arrival time of process 3: 0
Burst time of process 3: 4
Process_id of process 4: 2
Arrival time of process 4: 0
Burst time of process 4: 2
Process_id of process 5: 8
Arrival time of process 5: 0
Burst time of process 5: 2
Process_id of process 6: 6
Arrival time of process 6: 0
Burst time of process 6: 3
Processes arrival_time Burst time completion time Turn around time Waiting time
1 0 1 1 1 0
3 0 2 3 3 1
5 0 4 7 7 3
2 0 2 9 9 7
8 0 2 11 11 9
6 0 3 14 14 11
Average waiting time is : 5.166667
Average turn around time is: 7.500000
Process returned 0 (0x0) execution time : 16.204 s
Press any key to continue.

b) Different arrival time

CODEBOSS
K.Ram Mohan

FCFS

Q1) b) All arriving at different time

no:of process = 6

Input:

PID	AT	BT
1	1	1
5	5	4
3	7	2
2	1	2
8	10	2
6	8	3

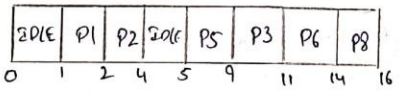
$CT[i] = BT[i] + AT[i]$ if $AT[i] > CT[i-1]$

$CT[i] = CT[i-1] + BT[i]$ else

OUTPUT:

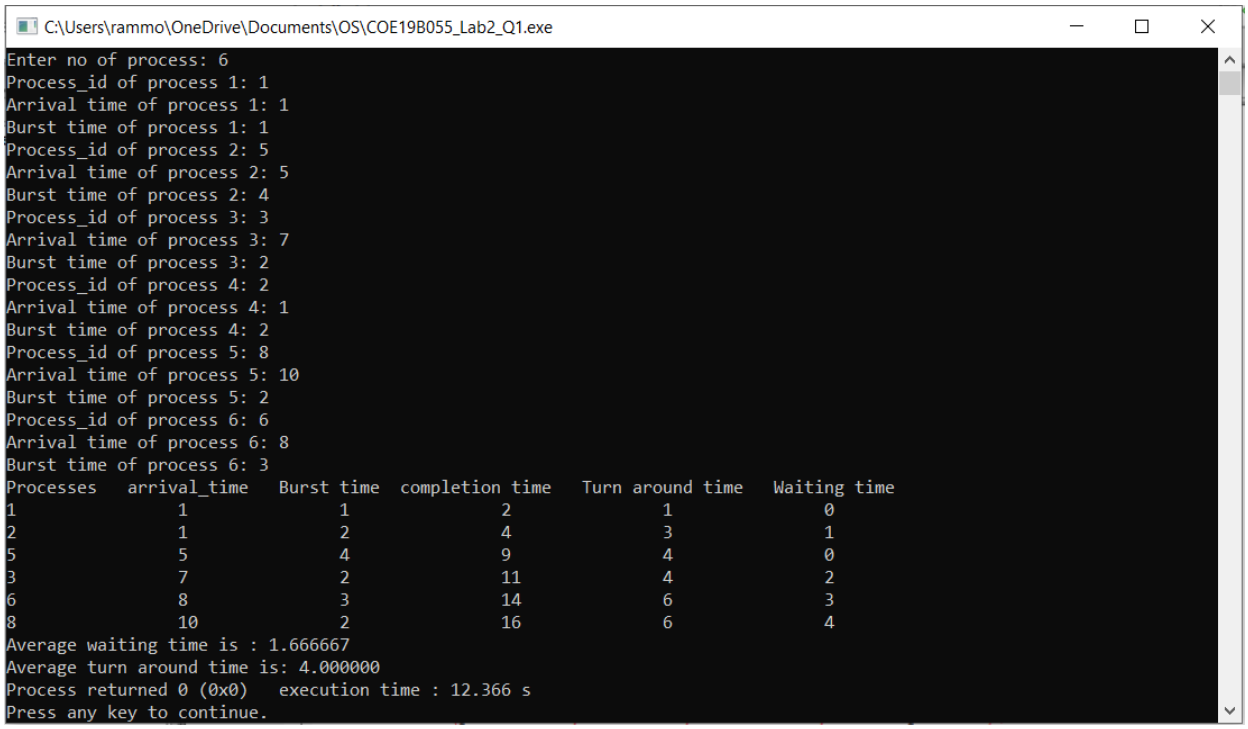
PID	AT	BT	CT	TAT	WT
1	1	1	2	1	0
2	1	2	4	3	1
5	5	4	9	4	0
3	7	2	11	4	2
6	8	3	14	6	3
8	10	2	16	6	4

GANTT CHART :



Average waiting time = 1%

Average turnaround time = 4


C:\Users\rammo\OneDrive\Documents\OS\COE19B055_Lab2_Q1.exe
Enter no of process: 6
Process_id of process 1: 1
Arrival time of process 1: 1
Burst time of process 1: 1
Process_id of process 2: 5
Arrival time of process 2: 5
Burst time of process 2: 4
Process_id of process 3: 3
Arrival time of process 3: 7
Burst time of process 3: 2
Process_id of process 4: 2
Arrival time of process 4: 1
Burst time of process 4: 2
Process_id of process 5: 8
Arrival time of process 5: 10
Burst time of process 5: 2
Process_id of process 6: 6
Arrival time of process 6: 8
Burst time of process 6: 3
Processes arrival_time Burst time completion time Turn around time Waiting time
1 1 1 2 1 0
2 1 2 4 3 1
5 5 4 9 4 0
3 7 2 11 4 2
6 8 3 14 6 3
8 10 2 16 6 4
Average waiting time is : 1.666667
Average turn around time is: 4.000000
Process returned 0 (0x0) execution time : 12.366 s
Press any key to continue.

Q2) SJF

```
#include<stdio.h>
#include<stdlib.h>
#define SIZE 100

void swap(int *p,int *q)
{
   int temp;
   temp=*p;
   *p=*q;
   *q=temp;
}
 //function to sort based on arrival time
void sort_arr(int id[], int arr[], int burst_time[] , int size)
{
   int i, j;
   int go=1;
   while(go)
   {
      for(i=0; i<size-1; i++)
      {
         go=0;
         for(j=0; j<size-1-i; j++)
         {
            if(arr[j]>arr[j+1])
            {
               swap(&arr[j], &arr[j+1]);
               swap(&id[j], &id[j+1]);
```

```
                swap(&burst_time[j], &burst_time[j+1]);
                go=1;
            }
        }
    }
}

int main()
{
    int total_process, i, j;

    int process_id[SIZE], arr_time[SIZE], burst_time[SIZE], c_time=0, low_bt, var;
    float tot_waiting_time=0, tot_trt=0;

    printf("Enter no of process: ");
    scanf("%d", &total_process);

    for(i=0; i<total_process; i++)
    {
        printf("Process_id of process %d: ", i+1);
        scanf("%d", &process_id[i]);
        printf("Arrival time of process %d: ", i+1);
        scanf("%d", &arr_time[i]);
        printf("Burst time of process %d: ", i+1);
        scanf("%d", &burst_time[i]);
    }

    sort_arr(process_id, arr_time, burst_time, total_process);
```

```
  int completion_time[SIZE], waiting_time[SIZE], turn_arnd_time[SIZE];


  for(i=0; i<total_process; i++)
  {
     //condition is to allocate c_time for initial state and if cpu is in idle state
     c_time= ((i==0 || completion_time[i-1]<arr_time[i]) ? arr_time[i] : completion_time[i-
1]);
     low_bt = burst_time[i];
     printf("%d -c_time\n", c_time);
     printf("%d low\n", low_bt);
     for(j=i; j<total_process; j++)
     {
//condition is to check whether there are any process available with less burst time
        if(c_time>=arr_time[j] && low_bt>=burst_time[j])
        {            low_bt = burst_time[j];
           var = j;
        }
     }
     completion_time[var] = c_time + burst_time[var];
     printf("%d-compl %d-var", completion_time[var], var);
     swap(&process_id[var], &process_id[i]);
     swap(&arr_time[var], &arr_time[i]);
     swap(&burst_time[var], &burst_time[i]);
     swap(&completion_time[var], &completion_time[i]);
  }


  for(i=0; i<total_process; i++)
  {
     turn_arnd_time[i] = completion_time[i] - arr_time[i];
     tot_trt = tot_trt + turn_arnd_time[i];
```

```
    }

    for(i=0; i<total_process; i++)
    {
        waiting_time[i] = turn_arnd_time[i] - burst_time[i];
        tot_waiting_time = tot_waiting_time + waiting_time[i];
    }

    printf("Processes  arrival_time  Burst time  completion time  Turn around time  Waiting time\n");
    for(i=0; i<total_process; i++)
    {
        printf("%d", process_id[i]);
        printf("\t \t %d", arr_time[i]);
        printf("\t \t %d", burst_time[i]);
        printf("\t \t %d", completion_time[i]);
        printf("\t \t %d", turn_arnd_time[i]);
        printf("\t \t %d \n", waiting_time[i]);
    }
    float avg_wait = tot_waiting_time/total_process;
    float avg_trt = tot_trt/total_process;
    printf("Average waiting time is : %f\n", avg_wait);
    printf("Average turn around time is: %f", avg_trt);
}
```

a) All inputs with arrival time 0

COEI9B05S
K.Ram Mohan

Q2) a) Arrival time = 0 SJF

no:of process = 6

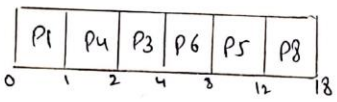
INPUT:

PID	AT	BT
1	0	1
5	0	4
3	0	2
4	0	1
6	0	4
8	0	6

OUTPUT:

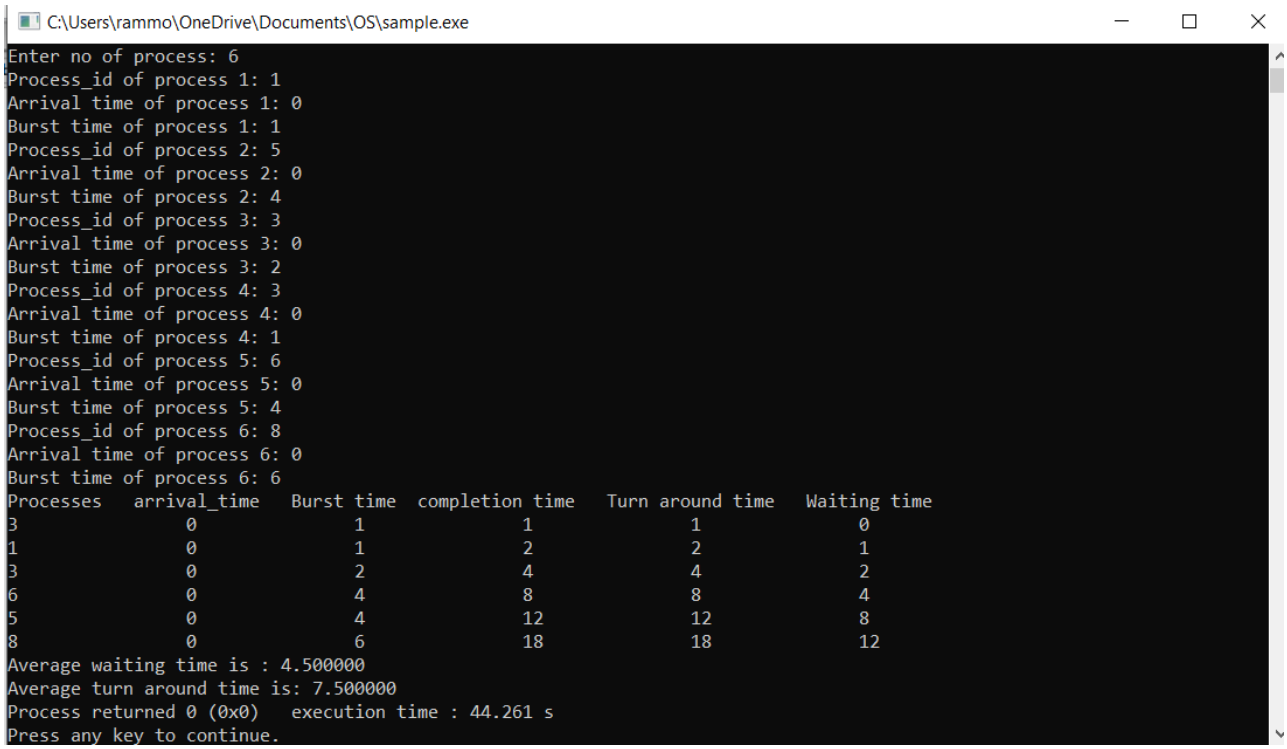
PID	AT	BT	CT	TAT	WT
1	0	1	1	1	0
4	0	1	2	2	1
3	0	2	4	4	2
6	0	4	8	8	4
5	0	4	12	12	8
8	0	6	18	18	12

GANTT CHART:



Average waiting time = 4.5

Average turn around time = 7.5


C:\Users\rammo\OneDrive\Documents\OS\sample.exe
Enter no of process: 6
Process_id of process 1: 1
Arrival time of process 1: 0
Burst time of process 1: 1
Process_id of process 2: 5
Arrival time of process 2: 0
Burst time of process 2: 4
Process_id of process 3: 3
Arrival time of process 3: 0
Burst time of process 3: 2
Process_id of process 4: 3
Arrival time of process 4: 0
Burst time of process 4: 1
Process_id of process 5: 6
Arrival time of process 5: 0
Burst time of process 5: 4
Process_id of process 6: 8
Arrival time of process 6: 0
Burst time of process 6: 6
Processes arrival_time Burst time completion time Turn around time Waiting time
3 0 1 1 1 0
1 0 1 2 2 1
3 0 2 4 4 2
6 0 4 8 8 4
5 0 4 12 12 8
8 0 6 18 18 12
Average waiting time is : 4.500000
Average turn around time is: 7.500000
Process returned 0 (0x0) execution time : 44.261 s
Press any key to continue.

b) Input with different arrival time

Q2) b

SJF

COE19CS055
K.Ram Mohan

All arriving at different time

No. of Process = 6

INPUT:

PID	AT	BT
1	6	4
2	2	5
3	1	3
4	1	1
5	4	2
6	1	6

Here we will check arrival time & burst time as well.

Also

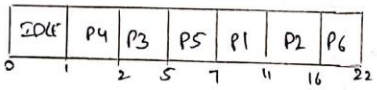
CT[i] = BT[i] + AT[i] if AT[i] > CT[i-1]

else CT[i] = CT[i-1] + BT[i-1]

OUTPUT

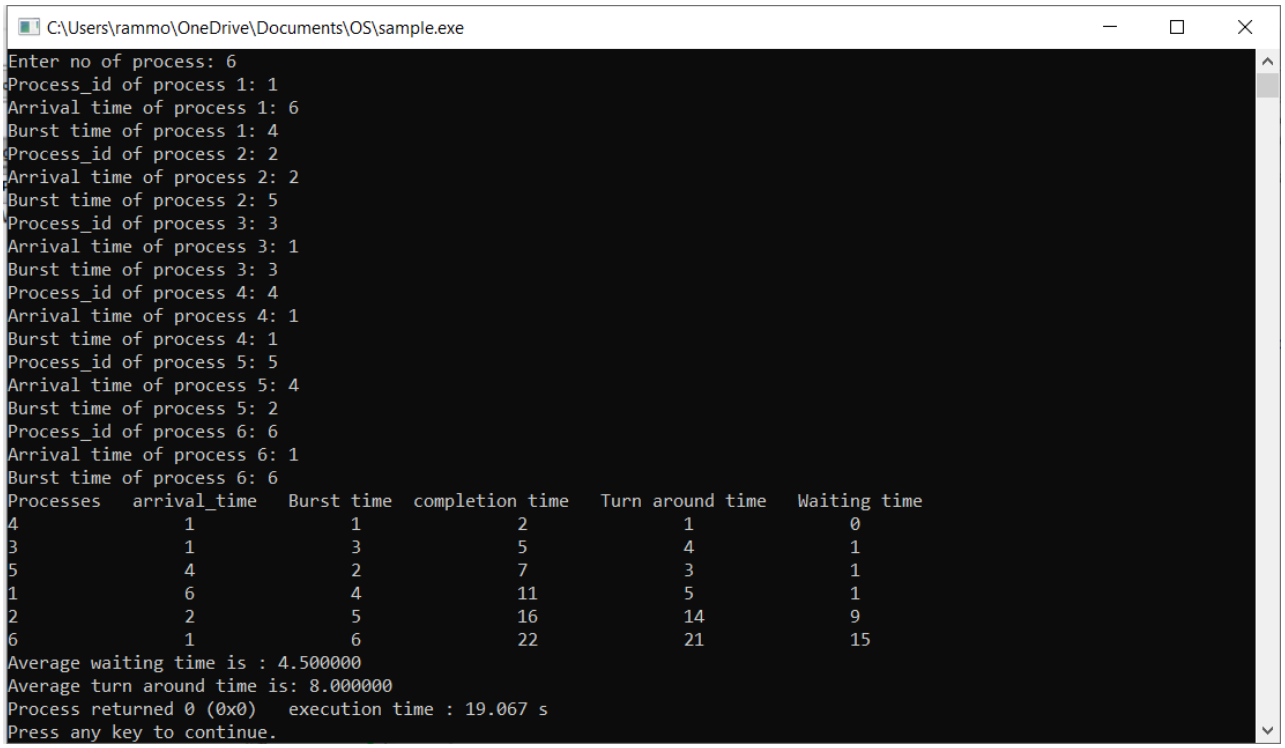
PID	AT	BT	CT	TAT	WT
4	1	1	2	1	0
3	1	3	5	4	1
5	4	2	7	3	1
1	6	4	11	5	1
2	2	5	16	14	9
6	1	6	22	21	15

GANTT CHART:



Average waiting time = 4.5

Average turn around time = 8


C:\Users\rammo\OneDrive\Documents\OS\sample.exe
Enter no of process: 6
Process_id of process 1: 1
Arrival time of process 1: 6
Burst time of process 1: 4
Process_id of process 2: 2
Arrival time of process 2: 2
Burst time of process 2: 5
Process_id of process 3: 3
Arrival time of process 3: 1
Burst time of process 3: 3
Process_id of process 4: 4
Arrival time of process 4: 1
Burst time of process 4: 1
Process_id of process 5: 5
Arrival time of process 5: 4
Burst time of process 5: 2
Process_id of process 6: 6
Arrival time of process 6: 1
Burst time of process 6: 6
Processes arrival_time Burst time completion time Turn around time Waiting time
4 1 1 2 1 0
3 1 3 5 4 1
5 4 2 7 3 1
1 6 4 11 5 1
2 2 5 16 14 9
6 1 6 22 21 15
Average waiting time is : 4.500000
Average turn around time is: 8.000000
Process returned 0 (0x0) execution time : 19.067 s
Press any key to continue.

Q3)

COE19B055
K.Ram Mohan

time :

Prints summary of real-time, user CPU time & system CPU time spent executing a command.

real time - time elapsed wall clock time taken by a command to get executed

user time - no:of CPUs seconds that command uses in user mode

sys time - " in kernal mode.

who :

It gives information about currently logged in user on to system

time whoami - gives username

time who - gives user name, some details

time who -r - displays current level of system

time w - list of users & their activities

time who -l -lt - login process details

time who -q -H - no:of users & name

```
ram@ram:~/Documents$ time whoami
ram

real    0m0.016s
user    0m0.000s
sys     0m0.006s
ram@ram:~/Documents$ time who
ram      :0           2021-08-15 10:08 (:0)

real    0m0.004s
user    0m0.004s
sys     0m0.001s
ram@ram:~/Documents$ time who -r
         run-level 5  2021-08-15 10:08

real    0m0.006s
user    0m0.004s
sys     0m0.001s
ram@ram:~/Documents$ time w
 11:47:55 up  1:40,  1 user,  load average: 0.35, 0.55, 0.50
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
ram      :0       :0               10:08   ?xdm?  17:43   0.06s /usr/lib/gdm3/gdm-x-session --ru

real    0m0.028s
user    0m0.005s
sys     0m0.015s
ram@ram:~/Documents$ time who -l -H
NAME     LINE         TIME             IDLE          PID COMMENT

real    0m0.005s
user    0m0.005s
sys     0m0.001s
ram@ram:~/Documents$ time who -q -H
ram
# users=1

real    0m0.005s
user    0m0.000s
sys     0m0.005s
ram@ram:~/Documents$ time who -a
           system boot  2021-08-15 10:07
           run-level 5  2021-08-15 10:08
ram      ? :0           2021-08-15 10:08   ?          1362 (:0)
```

```
ram@ram:~$ time;who>myfile

real    0m0.000s
user    0m0.000s
sys     0m0.000s
ram@ram:~$ more myfile
ram      :0           2021-08-15 19:25 (:0)
```

Q4)

a)tps:

4) PS:

CODEBOSS
K.Ram Mohan

PS - shows current process in the shell

PS -A - show all running process

PS -a - Processes not associated with terminal

PS -d - Process except session leaders

PS -a -N - view all process except those that fulfill specified conditions

PS -T - view all process associated with terminal

PS -r - view all running process

PS -x - view all process owned by you

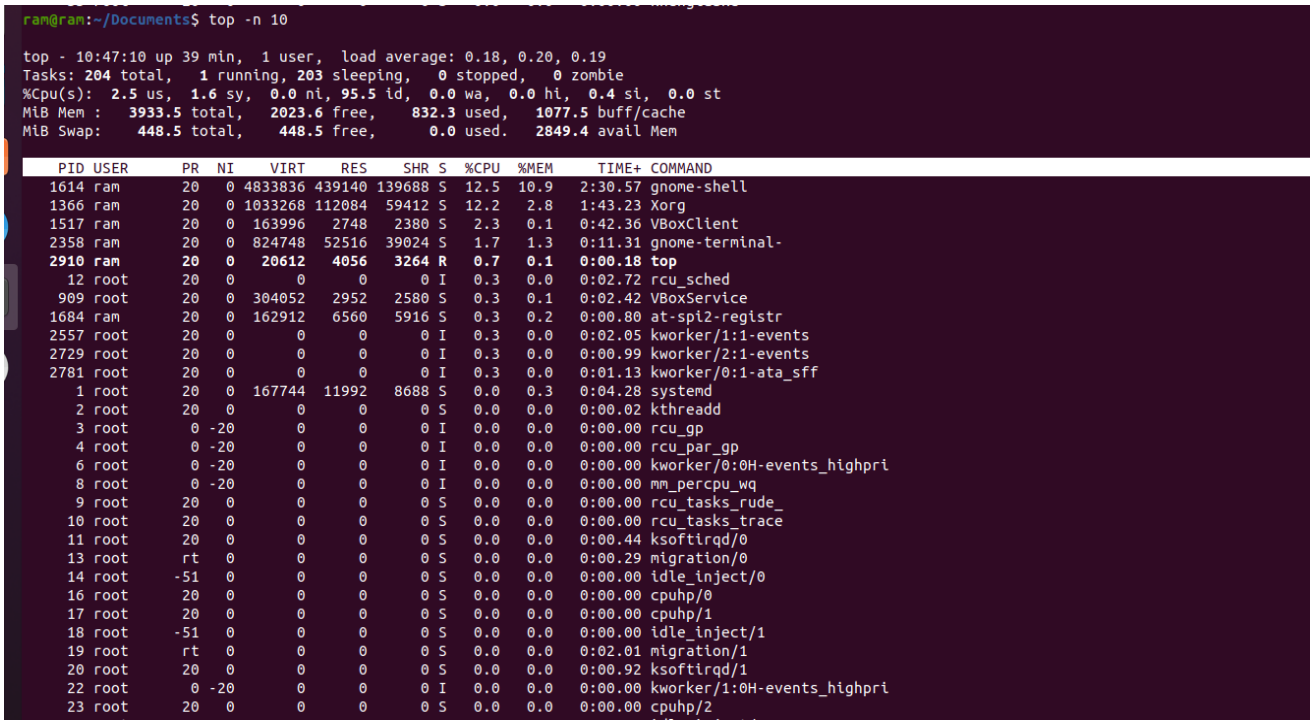

```
   5811    5811 pts/0    00:00:00 ps
ram@ram:~/Documents$ ps -r
    PID TTY      STAT   TIME COMMAND
   5817 pts/0    R+     0:00 ps -r
ram@ram:~/Documents$ ps
    PID TTY          TIME CMD
   2456 pts/0    00:00:00 bash
   5567 pts/0    00:00:10 firefox
   5637 pts/0    00:00:03 Privileged Cont
   5699 pts/0    00:00:00 Web Content
   5729 pts/0    00:00:01 WebExtensions
   5768 pts/0    00:00:00 Web Content
   5819 pts/0    00:00:00 ps
ram@ram:~/Documents$ ps -a
    PID TTY          TIME CMD
   1366 tty2     00:06:32 Xorg
   1426 tty2     00:00:00 gnome-session-b
   4919 pts/1    00:00:00 gedit
   5567 pts/0    00:00:10 firefox
   5637 pts/0    00:00:03 Privileged Cont
   5699 pts/0    00:00:00 Web Content
   5729 pts/0    00:00:01 WebExtensions
   5768 pts/0    00:00:00 Web Content
   5820 pts/0    00:00:00 ps
ram@ram:~/Documents$
```

B)top:

top: It shows Linux process

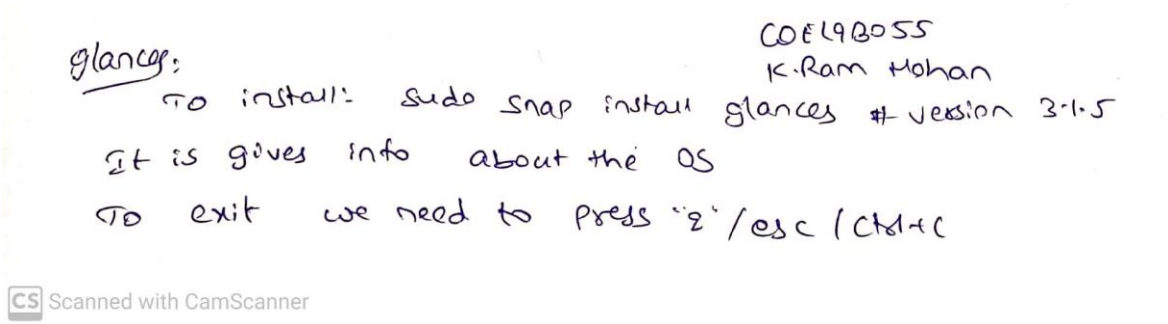
top -n 10 : gives 10 process

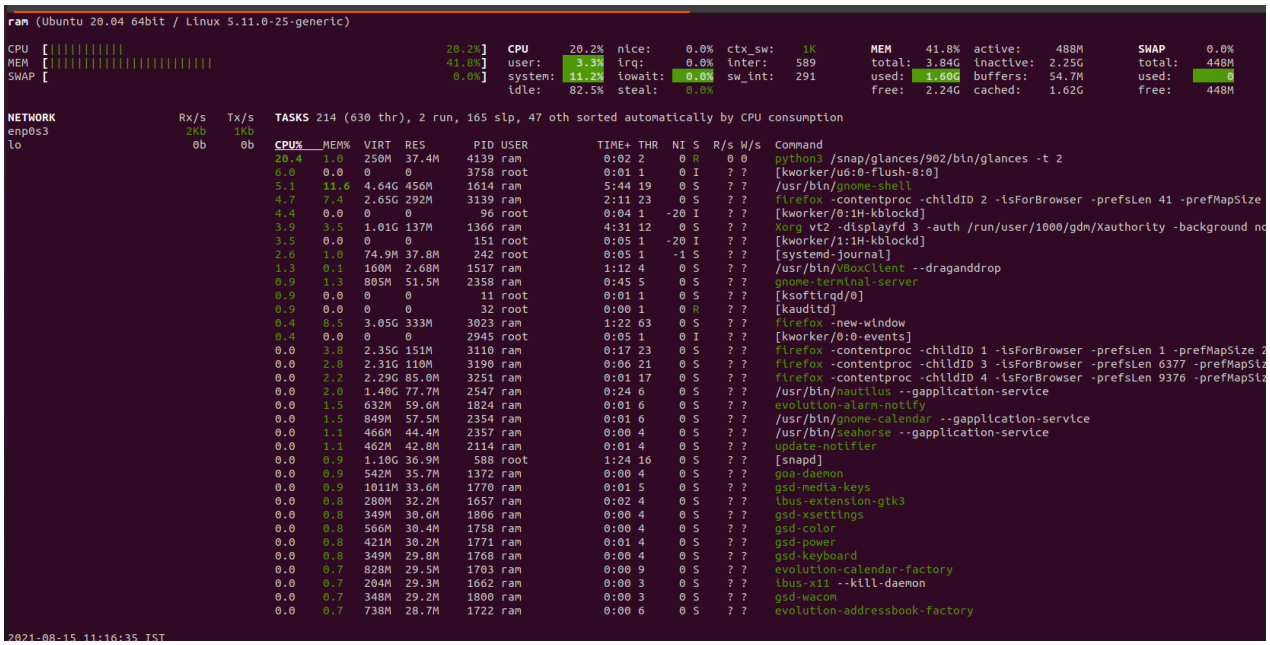
COE19B055
K.Ram Mohan



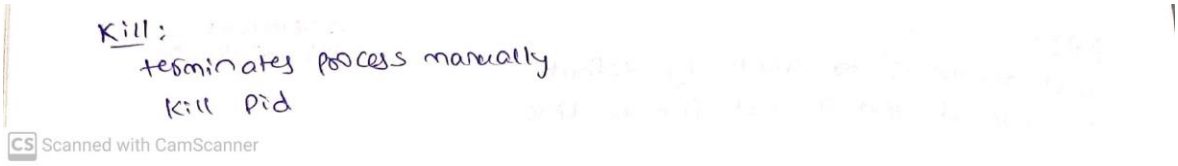
```
ram@ram:~/Documents$ top -n 10

top - 10:47:10 up 39 min,  1 user,  load average: 0.18, 0.20, 0.19
Tasks: 204 total,   1 running, 203 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.5 us,  1.6 sy,  0.0 ni, 95.5 id,  0.0 wa,  0.0 hi,  0.4 si,  0.0 st
MiB Mem :   3933.5 total,   2023.6 free,    832.3 used,   1077.5 buff/cache
MiB Swap:    448.5 total,    448.5 free,      0.0 used.   2849.4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1614 ram       20   0 4833836 439140 139688 S  12.5  10.9   2:30.57 gnome-shell
   1366 ram       20   0 1033268 112084  59412 S  12.2   2.8   1:43.23 Xorg
   1517 ram       20   0  163996   2748   2380 S   2.3   0.1   0:42.36 VBoxClient
   2358 ram       20   0  824748  52516  39024 S   1.7   1.3   0:11.31 gnome-terminal-
   2910 ram       20   0   20612   4056   3264 R   0.7   0.1   0:00.18 top
     12 root      20   0       0      0      0 I   0.3   0.0   0:02.72 rcu_sched
    909 root      20   0  304052   2952   2580 S   0.3   0.1   0:02.42 VBoxService
   1684 ram       20   0  162912   6560   5916 S   0.3   0.2   0:00.80 at-spi2-registr
   2557 root      20   0       0      0      0 I   0.3   0.0   0:02.05 kworker/1:1-events
   2729 root      20   0       0      0      0 I   0.3   0.0   0:00.99 kworker/2:1-events
   2781 root      20   0       0      0      0 I   0.3   0.0   0:01.13 kworker/0:1-ata_sff
      1 root      20   0  167744  11992   8688 S   0.0   0.3   0:04.28 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.02 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
      8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
      9 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_rude_
     10 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_trace
     11 root      20   0       0      0      0 S   0.0   0.0   0:00.44 ksoftirqd/0
     13 root      rt   0       0      0      0 S   0.0   0.0   0:00.29 migration/0
     14 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/0
     16 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     17 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
     18 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/1
     19 root      rt   0       0      0      0 S   0.0   0.0   0:02.01 migration/1
     20 root      20   0       0      0      0 S   0.0   0.0   0:00.92 ksoftirqd/1
     22 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/1:0H-events_highpri
     23 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/2
```

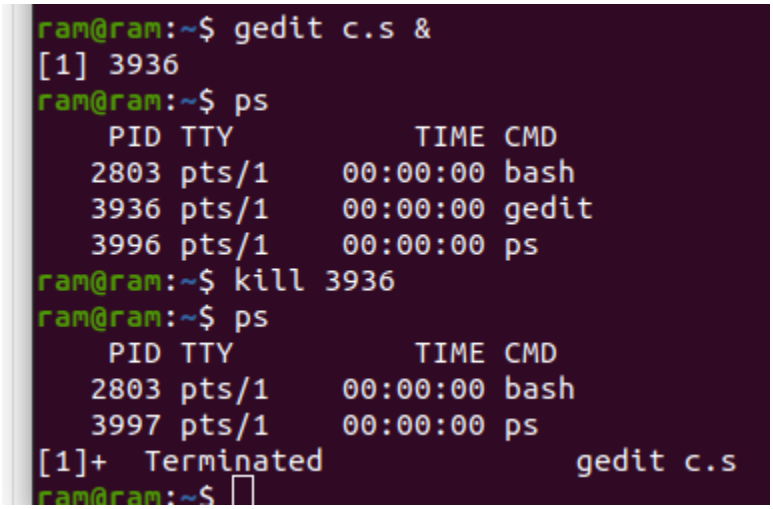
C)glances:



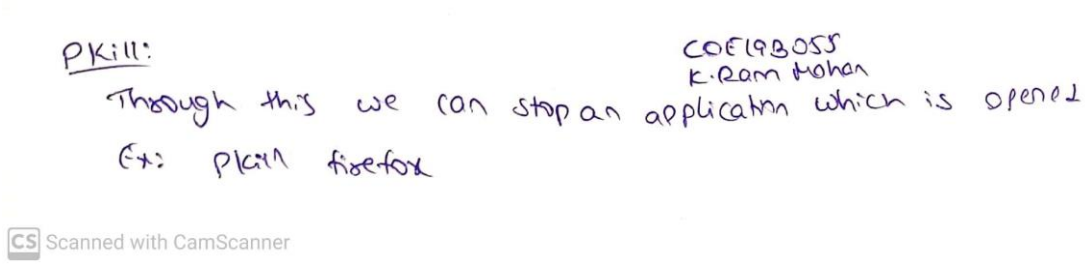


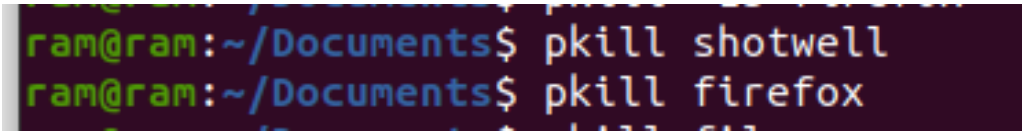
D)Kill:



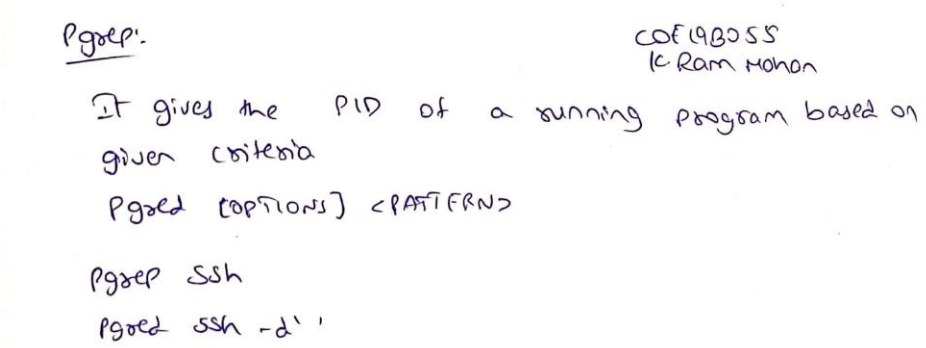


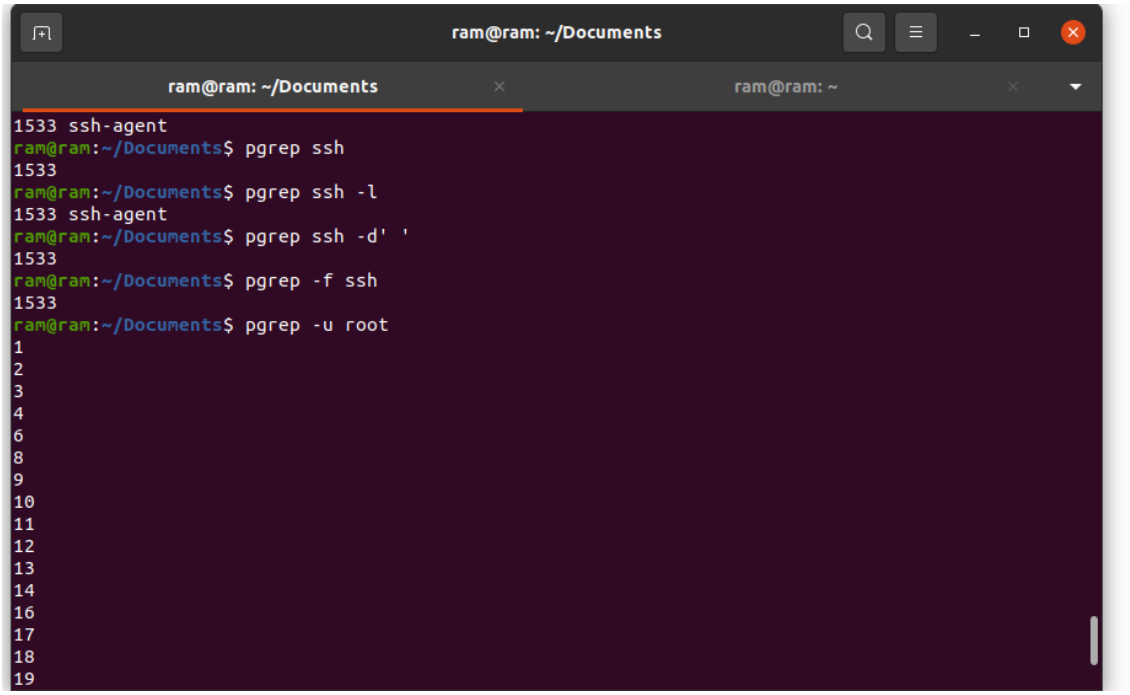
E)Pkill:





F)pgrep:





Extra:

Select:

It is used to create a numbered menu from which a user can select an option. If user enter valid option the it executes set of commands written in select block and ask again to enter again.

If a user didn't press any option & pressed enter then it shows list of options.


```
bash: syntax error near unexpected token `sub=$(( a-b ))'
ram@ram:~/Documents$ a=10
ram@ram:~/Documents$ b=5
ram@ram:~/Documents$ select i in Addition Subtraction Multiplication Division; do case $i in Add
ition) add=$(( a+b )); echo $add;; Subtraction) sub=$(( a-b ))
> echo $sub;;
> Multiplication) mul=$(( a*b ))
> echo $mul;;
> Division) div=$(( a/b ))
> echo $div;;
> esac
> done
1) Addition
2) Subtraction
3) Multiplication
4) Division
#? 1
15
#? 2
5
#? 3
50
#? 4
2
#?
```