EXCERCISE 1

1.Create a new process by invoking the appropriate system call. Get the process identifier of the currently running process and its respective parent using system calls and display the same using a C program.

```
#include<stdio.h>
#include<unistd.h>
int main()
{
  printf("Process ID: %d\n", getpid() );
  printf("Parent Process ID: %d\n", getpid() );
  return 0;
}
```

```
Process ID: 1916
Parent Process ID: 1916

--------------------------------
Process exited after 0.03943 seconds with return value 0
Press any key to continue . . .
```

2.Identify the system calls to copy the content of one file to another and illustrate The same using a C program.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *fptr1, *fptr2;
    char filename[100], c;
    printf("Enter the filename to open for reading \n");
    scanf("%s", filename);
    fptr1 = fopen(filename, "r");
    if (fptr1 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }
    printf("Enter the filename to open for writing \n");
    scanf("%s", filename);
    fptr2 = fopen(filename, "w");
    if (fptr2 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }
    c = fgetc(fptr1);
    while (c != EOF)
    {
        fputc(c, fptr2);
        c = fgetc(fptr1);
    }
    printf("\nContents copied to %s", filename);
    fclose(fptr1);
    fclose(fptr2);
    return 0;
}
```

```
Enter the filename to open for reading
os.2.cpp
Enter the filename to open for writing
swetha

Contents copied to swetha
--------------------------------
Process exited after 22.85 seconds with return value 0
Press any key to continue . . .
```

EXCERCISE 1

3. Design a CPU scheduling program with C using First Come First Served technique with the following considerations.

a. All processes are activated at time 0.

b. Assume that no process waits on I/O devices.

```
≡ File Edit Search Run Compile Debug Project Options Window Help
[■]                    SWETHA3.CPP                    1-[↕]
#include <stdio.h>
int main()
{
        int A[100][4];
        int i, j, n, total = 0, index, temp;
        float avg_wt, avg_tat;
        printf("Enter number of process: ");
        scanf("%d", &n);
        printf("Enter Burst Time:\n");
        for (i = 0; i < n; i++) {
                printf("P%d: ", i + 1);
                scanf("%d", &A[i][1]);
                A[i][0] = i + 1;
        }
        for (i = 0; i < n; i++) {
                index = i;
                for (j = i + 1; j < n; j++)
                        if (A[j][1] < A[index][1])
                                index = j;
                temp = A[i][1];
                A[i][1] = A[index][1];
        49:88
F1 Help  F2 Save  F3 Open  Alt-F9 Compile  F9 Make  F10 Menu
```

Output

```
C:\TURBOC3\BIN>TC
Enter number of process: 3
Enter Burst Time:
P1: 1
P2: 2
P3: 3
P        BT       WT       TAT
P1       1        0        1
P2       2        1        3
P3       3        3        6
Average Waiting Time= 1.333333
Average Turnaround Time= 3.333333Enter number of process:
```

EXCERCISE 1

4. Construct a scheduling program with C that selects the waiting process with the smallest execution time to execute next.

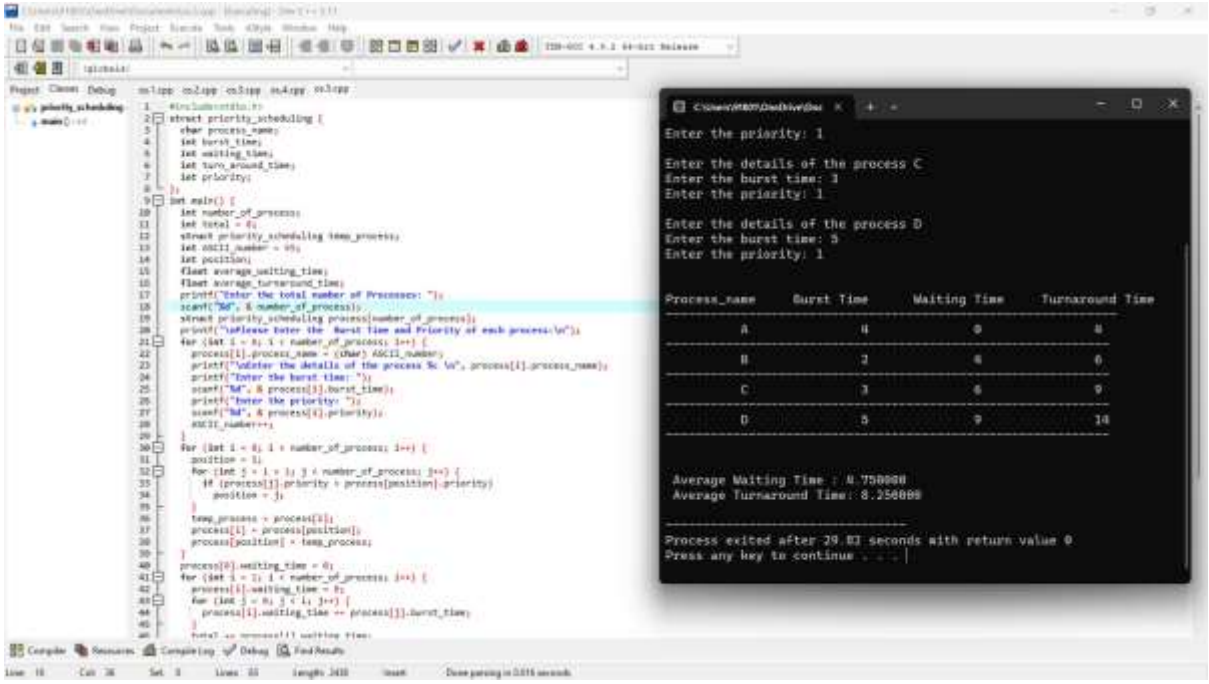
```
≡ File Edit Search Run Compile Debug Project Options Window Help
[■]                    SWETHA4.CPP                    2-[↕]
#include<stdio.h>
 int main()
{
    int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
    float avg_wt,avg_tat;
    printf("Enter number of process:");
    scanf("%d",&n);
    printf("nEnter Burst Time:n");
    for(i=0;i<n;i++)
    {
        printf("p%d:",i+1);
        scanf("%d",&bt[i]);
        p[i]=i+1;
    }
    for(i=0;i<n;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
            if(bt[j]<bt[pos])
                pos=j;
1:38
F1 Help  F2 Save  F3 Open  Alt-F9 Compile  F9 Make  F10 Menu
```

Output

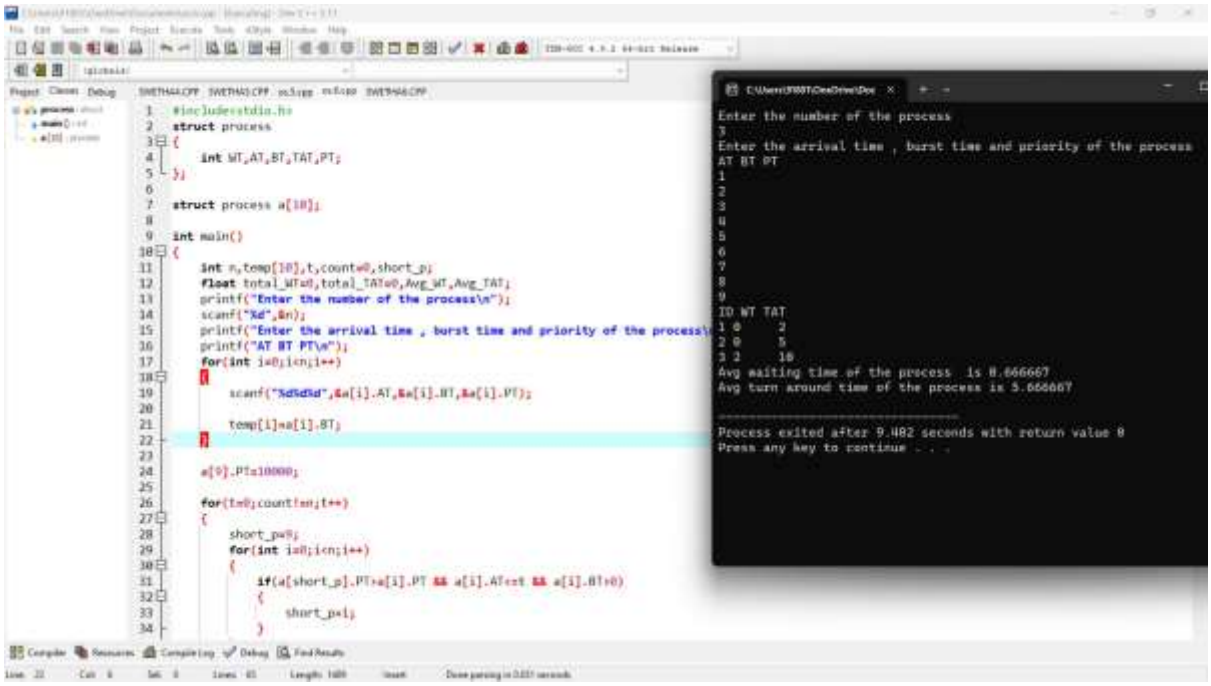
```
C:\TURBOC3\BIN>TC
Enter number of process:3
nEnter Burst Time:np1:6
p2:7
p3:8
nProcesst    Burst Time    tWaiting TimetTurnaround Timenp1tt  6tt    0ttt6np2tt
  7tt    6ttt13np3tt  8tt    13ttt21nnAverage Waiting Time=6.333333nAverage Turn
around Time=13.333333nEnter number of process:
```

EXCERCISE 1

5. Construct a scheduling program with C that selects the waiting process with the highest priority to execute next.



6. Construct a C program to implement pre-emptive priority scheduling algorithm.



EXCERCISE 1

7. Construct a C program to implement non-preemptive SJF algorithm.

```
≡ File Edit Search Run Compile Debug Project Options Window Help
[■]                    SWETHA7C.CPP                    6-[↕]
#include<stdio.h>

int main() {
  int time, burst_time[10], at[10], sum_burst_time = 0, smallest, n, i;
  int sumt = 0, sumw = 0;
  printf("enter the no of processes : ");
  scanf("%d", & n);
  for (i = 0; i < n; i++) {
    printf("the arrival time for process P%d : ", i + 1);
    scanf("%d", & at[i]);
    printf("the burst time for process P%d : ", i + 1);
    scanf("%d", & burst_time[i]);
    sum_burst_time += burst_time[i];
  }
  burst_time[9] = 9999;
  for (time = 0; time < sum_burst_time;) {
    smallest = 9;
    for (i = 0; i < n; i++) {
      if (at[i] <= time && burst_time[i] > 0 && burst_time[i] < burst_time[sma
        smallest = i;
    }
1:8
F1 Help  Alt-F8 Next Msg  Alt-F7 Prev Msg  Alt-F9 Compile  F9 Make  F10 Menu
```

Output

```
C:\TURBOC3\BIN>TC
enter the no of processes : 2
the arrival time for process P1 : 1
the burst time for process P1 : 2
the arrival time for process P2 : 4
the burst time for process P2 : 5
P[10]   |       9999    |       0


 average waiting time = 0.000000

 average turnaround time = 4999.500000enter the no of processes : 1
the arrival time for process P1 : 1
the burst time for process P1 : 2
P[10]   |       8290    |       -1709


 average waiting time = -1709.000000

 average turnaround time = 8290.000000enter the no of processes : _
```

EXCERCISE 1

8. Construct a C program to simulate Round Robin scheduling algorithm with C.

```
≡ File Edit Search Run Compile Debug Project Options Window Help
[■]                    SWETHA8.CPP                    7-[↕]
#include<stdio.h>
#include<conio.h>
int main()
{
    int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10]
    float avg_wt, avg_tat;
    printf(" Total number of process in the system: ");
    scanf("%d", &NOP);
    y = NOP;
for(i=0; i<NOP; i++)
{
printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);
printf(" Arrival time is: \t");
scanf("%d", &at[i]);
printf(" \nBurst time is: \t");
scanf("%d", &bt[i]);
temp[i] = bt[i];
}
printf("Enter the Time Quantum for the process: \t");
scanf("%d", &quant);
printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");
        1:8
F1 Help  F2 Save  F3 Open  Alt-F9 Compile  F9 Make  F10 Menu
```

Output

```
C:\TURBOC3\BIN>TC
 Total number of process in the system: 3

 Enter the Arrival and Burst time of the Process[1]
 Arrival time is:       12

Burst time is:  10

 Enter the Arrival and Burst time of the Process[2]
 Arrival time is:       11

Burst time is:  1

 Enter the Arrival and Burst time of the Process[3]
 Arrival time is:       11

Burst time is:  2
Enter the Time Quantum for the process:         10

 Process No              Burst Time              TAT             Waiting Time
Process No[1]            10                              -2
 -12
```

EXCERCISE 1

9. Design a C program to simulate the concept of Dining-Philosophers problem

```c
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<semaphore.h>
#include<unistd.h>
sem_t room;
sem_t chopstick[5];
void * philosopher(void *);
void eat(int);
int main()
{
    int i,a[5];
    pthread_t tid[5];
    sem_init(&room,0,4);
    for(i=0;i<5;i++)
        sem_init(&chopstick[i],0,1);
    for(i=0;i<5;i++){
        a[i]=i;
        pthread_create(&tid[i],NULL,philosopher,(void *)&a[i]);
    }
    for(i=0;i<5;i++)
        pthread_join(tid[i],NULL);
}
void * philosopher(void * num)
{
    int phil=*(int *)num;
    sem_wait(&room);
    printf("\nPhilosopher %d has entered room",phil);
    sem_wait(&chopstick[phil]);
    sem_wait(&chopstick[(phil+1)%5]);
    eat(phil);
    sleep(2);
    printf("\nPhilosopher %d has finished eating",phil);
    sem_post(&chopstick[(phil+1)%5]);
```

Output

```
Philosopher 2 has entered room
Philosopher 2 is eating
Philosopher 1 has entered room
Philosopher 3 has entered room
Philosopher 2 has finished eating
Philosopher 3 is eating
Philosopher 0 has finished eating
Philosopher 4 has entered room
Philosopher 1 is eating
Philosopher 3 has finished eating
Philosopher 1 has finished eating
Philosopher 4 is eating
Philosopher 4 has finished eating
--------------------------------
Process exited after 6.104 seconds with return value 0
Press any key to continue . . .
```

EXCERCISE 1

10. Construct a C program for implementation of memory allocation using first fit strategy.

```c
#include<stdio.h>
int main()
{
    int bsize[10], psize[10], bno, pno, flags[10], allocation[10], i, j;
    for(i = 0; i < 10; i++)
    {
        flags[i] = 0;
        allocation[i] = -1;
    }
    printf("Enter no. of blocks: ");
    scanf("%d", &bno);
    printf("\nEnter size of each block: ");
    for(i = 0; i < bno; i++)
        scanf("%d", &bsize[i]);
    printf("\nEnter no. of processes: ");
    scanf("%d", &pno);
    printf("\nEnter size of each process: ");
    for(i = 0; i < pno; i++)
        scanf("%d", &psize[i]);
    for(i = 0; i < pno; i++)
        for(j = 0; j < bno; j++)
            if(flags[j] == 0 && bsize[j] >= psize[i])
            {
                allocation[j] = i;
                flags[j] = 1;
                break;
            }
    printf("\nBlock no.\tsize\t\tprocess no.\t\tsize");
    for(i = 0; i < bno; i++)
    {
        printf("\n%d\t\t%d\t\t", i+1, bsize[i]);
        if(flags[i] == 1)
            printf("%d\t\t\t%d",allocation[i]+1,psize[allocation[i]]);
        else
            printf("Not allocated");
    }
}
```

Output

```
Enter no. of blocks: 2

Enter size of each block: 10
20

Enter no. of processes: 1

Enter size of each process: 30

Block no.       size            process no.             size
1               10              Not allocated
2               20              Not allocated
--------------------------------
Process exited after 15.64 seconds with return value 0
Press any key to continue . . .
```