Registration Number: 19BCE2119

Name: Gaurav Kumar Singh

Course: Operating Systems

Cycle sheet Q7-14

7) Write a C program to kill a process by specifying its name rather than its PID.

CODE

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
int main() {
        printf("\nList of processes:\n");
        system("ps -aux");
        char command[]="pkill -9 ";
        char process_name[100];
        printf("\nEnter the name of the process to be killed:");
        scanf("%s",&process_name);
        char P[100];
        strcpy (P,process_name);
        strcat(command,process_name);
        printf("\n");
        system (command);
        printf("\nList of processes after killing '%s' process.\n",P);
        system("ps -aux");

    return 0;
}
```

OUTPUT

Select gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
int main() {
                printf("\nList of processes:\n");
                system("ps -aux");
                char command[]="pkill -9 ";
                char process_name[100];
                printf("\nEnter the name of the process to be killed:");
                scanf("%s",&process_name);
                char P[100];
                strcpy (P,process_name);
                strcat(command,process_name);
                printf("\n");
                system (command);
                printf("\nList of processes after killing '%s' process.\n",P);
                system("ps -aux");

        return 0;
}
~
```

gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2

```
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc killByName.c -o killByName
killByName.c: In function ‘main’:
killByName.c:11:11: warning: format ‘%s’ expects argument of type ‘char *’, but argument 2 has type ‘char (*)[100]’ [-Wformat=]
   11 |   scanf("%s",&process_name);
      |          ~^   ~~~~~~~~~~~~~
      |           |   |
      |           |   char (*)[100]
      |           char *
killByName.c:13:3: warning: implicit declaration of function ‘strcpy’ [-Wimplicit-function-declaration]
   13 |   strcpy (P,process_name);
      |   ^~~~~~
killByName.c:13:3: warning: incompatible implicit declaration of built-in function ‘strcpy’
killByName.c:5:1: note: include ‘<string.h>’ or provide a declaration of ‘strcpy’
    4 | #include <stdlib.h>
  +++ |+#include <string.h>
    5 | int main() {
killByName.c:14:3: warning: implicit declaration of function ‘strcat’ [-Wimplicit-function-declaration]
   14 |   strcat(command,process_name);
      |   ^~~~~~
killByName.c:14:3: warning: incompatible implicit declaration of built-in function ‘strcat’
killByName.c:14:3: note: include ‘<string.h>’ or provide a declaration of ‘strcat’
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ sleep 100 &
[1] 47
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./killByName

List of processes:
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   8940   320 ?        Ssl  09:30   0:00 /init
root         8  0.0  0.0   8940   228 tty1     Ss   09:30   0:00 /init
gaurav1+     9  0.0  0.0  18208  3676 tty1     S    09:30   0:00 -bash
gaurav1+    47  0.1  0.0  15276   820 tty1     S    09:37   0:00 sleep 100
gaurav1+    48  0.0  0.0  10536   568 tty1     S    09:37   0:00 ./killByName
gaurav1+    49  0.0  0.0  10656   688 tty1     S    09:37   0:00 sh -c ps -aux
gaurav1+    50  0.0  0.0  18880  2024 tty1     R    09:37   0:00 ps -aux

Enter the name of the process to be killed:sleep


List of processes after killing 'sleep' process.
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   8940   320 ?        Ssl  09:30   0:00 /init
root         8  0.0  0.0   8940   228 tty1     Ss   09:30   0:00 /init
gaurav1+     9  0.0  0.0  18208  3676 tty1     S    09:30   0:00 -bash
gaurav1+    48  0.0  0.0  10536   628 tty1     S    09:37   0:00 ./killByName
gaurav1+    53  0.0  0.0  10656   688 tty1     S    09:37   0:00 sh -c ps -aux
gaurav1+    54  0.0  0.0  18880  2024 tty1     R    09:37   0:00 ps -aux
[1]+  Killed                  sleep 100
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
```

8) Create a file with few lines. Write a C program to read the file and delete the spaces more than one in the file (use UNIX file API's).

CODE

```
#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

#include <stdlib.h>

#include <fcntl.h>

#include <string.h>

void remSpace(char *str,char *retstr){

	int i;

	int counter=0;

	for (i=0;str[i];i++){

		if (str[i]!=' '){

			retstr[counter]=str[i];

			counter++;

		}

		else if (str[i]==' '){

			retstr[counter]=' ';

			while(str[i+1]==' '){

				i++;

			}

			counter++;

		}

	}

}

int main() {

	int fd;

	char buffer[80];

	char retbuffer[80];

	fd=open("test.txt",O_RDWR);
```

```
    printf("fd=%d",fd);
    if (fd!=-1){
        printf("\ntest.txt opened with read and write access.\n");
        lseek(fd,0,SEEK_SET);
        read(fd,buffer,50);
        printf("\nText inside the Document is:%s",buffer);
        remSpace(buffer,retbuffer);
        lseek(fd,0,SEEK_SET);
        write(fd,retbuffer,sizeof(retbuffer));
        lseek(fd,0,SEEK_SET);
        read(fd,buffer,sizeof(retbuffer));
        printf("\nReading text from the file after removing more than one spaces: %s",buffer);
        printf("\n\n");
        close(fd);
    }
return 0;

}
```

OUTPUT

gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
void remSpace(char *str,char *retstr){
        int i;
        int counter=0;
        for (i=0;str[i];i++){
                if (str[i]!=' '){
                        retstr[counter]=str[i];
                        counter++;
                }
                else if (str[i]==' '){
                        retstr[counter]=' ';
                        while(str[i+1]==' '){
                                i++;
                        }
                        counter++;
                }
        }
}
int main() {
        int fd;
        char buffer[80];
        char retbuffer[80];
        fd=open("test.txt",O_RDWR);
        printf("fd=%d",fd);
        if (fd!=-1){
                printf("\ntest.txt opened with read and write access.\n");
                lseek(fd,0,SEEK_SET);
                read(fd,buffer,50);
                printf("\nText inside the Document is:%s",buffer);
                remSpace(buffer,retbuffer);
                lseek(fd,0,SEEK_SET);
                write(fd,retbuffer,sizeof(retbuffer));
                lseek(fd,0,SEEK_SET);
                read(fd,buffer,sizeof(retbuffer));
                printf("\nReading text from the file after removing more than one spaces: %s",buffer);
                printf("\n\n");
                close(fd);
        }
return 0;

}
```

gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2

```
Hello World.      How   are   you          doing?
~
~
~
~
```

gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2

```
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc remSpace.c -o remSpace
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./remSpace
fd=3
test.txt opened with read and write access.

Text inside the Document is:Hello World.      How   are   you          doing?

Reading text from the file after removing more than one spaces: Hello World. How are you doing?
□□

gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
```

9) Write a program

a) . To create parent & child process and print their id.

CODE

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>



int main() {

	pid_t pid;

	pid=fork();

	if (pid==0){

		printf("\n(i)\n\n");

		printf("\nChild process created with Process ID: %d",getpid());

		printf("\nChild process created with parent process ID: %d",getppid());

		exit(0);

	}

	else if(pid>0) {

		wait(0);

		printf("\nParent process created with Process ID: %d",getpid());

		printf("\n");

	}

	return 0;

}
```

OUTPUT

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>


int main() {
        pid_t pid;
        pid=fork();
        if (pid==0){
                printf("\n(i)\n\n");
                printf("\nChild process created with Process ID: %d",getpid());
                printf("\nChild process created with parent process ID: %d",getppid());
                exit(0);
        }
        else if(pid>0) {
                wait(0);
                printf("\nParent process created with Process ID: %d",getpid());
                printf("\n");
        }
        return 0;
}
~
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc Parent_Child.c -o Parent_Child
Parent_Child.c: In function ‘main’:
Parent_Child.c:13:17: warning: implicit declaration of function ‘exit’ [-Wimplicit-function-declaration]
   13 |                 exit(0);
      |                 ^~~~
Parent_Child.c:13:17: warning: incompatible implicit declaration of built-in function ‘exit’
Parent_Child.c:4:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
    3 | #include <unistd.h>
  +++ |+#include <stdlib.h>
    4 |
Parent_Child.c:16:3: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-declaration]
   16 |   wait(0);
      |   ^~~~
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./Parent_Child

(i)


Child process created with Process ID: 97
Child process created with parent process ID: 96
Parent process created with Process ID: 96
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
```

b) . To create a zombie process.

CODE

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>



int main() {

        pid_t pid;
```

```
    pid=fork();
    if (pid==0){
        printf("\n(ii)\n\n");
        printf("\nThis is Child Process with process ID: %d",getpid());
        printf("\nParent Process ID of this child process is: %d",getppid());
        printf("\nThe child process is now converted to a zombie process because its parent process is alive.");
        exit(0);
    }
    else if(pid>0){
        sleep(1);
        printf("\nParent process id is:%d",getpid());
    }
    return 0;
}
```

OUTPUT

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>


int main() {
        pid_t pid;
        pid=fork();
        if (pid==0){
                printf("\n(iii)\n\n");
                printf("\nThis is Child Process with process ID: %d",getpid());
                printf("\nParent Process ID of this child process is: %d",getppid());
                printf("\nThe child process is now converted to a zombie process because its parent process is alive.");
                exit(0);
        }
        else if(pid>0){
                sleep(1);
                printf("\nParent process id is:%d",getpid());
        }
        return 0;
}
~
```

```
Select gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc Zombie.c -o Zombie
Zombie.c: In function ‘main’:
Zombie.c:14:17: warning: implicit declaration of function ‘exit’ [-Wimplicit-function-declaration]
   14 |                 exit(0);
      |                 ^~~~
Zombie.c:14:17: warning: incompatible implicit declaration of built-in function ‘exit’
Zombie.c:4:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
    3 | #include <unistd.h>
  +++ |+#include <stdlib.h>
    4 |
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./Zombie

(ii)


This is Child Process with process ID: 116
Parent Process ID of this child process is: 115
The child process is now converted to a zombie process because its parent process is alive.
Parent process id is:115gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
```

c) To create orphan process

CODE

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>


int main() {

    pid_t pid;

    pid=fork();

    if (pid==0){

        sleep(1);

        printf("\nChild process created with Process ID: %d",getpid());

        printf("\nChild process created with parent process ID: %d",getppid());

        printf("\nThis is an Orphan process now");

    }

    else {

        printf("\n(iii)\n\n");

        printf("\nParent process has Process ID: %d",getpid());

    }

    return 0;

}
```

OUTPUT

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>


int main() {
        pid_t pid;
        pid=fork();
        if (pid==0){
                sleep(1);
                printf("\nChild process created with Process ID: %d",getpid());
                printf("\nChild process created with parent process ID: %d",getppid());
                printf("\nThis is an Orphan process now");
        }
        else {
                printf("\n(iii)\n\n");
                printf("\nParent process has Process ID: %d",getpid());
        }
        return 0;
}
~
~
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc Orphan.c -o Orphan
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./Orphan

(iii)


Parent process has Process ID: 135gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
Child process created with Process ID: 136
Child process created with parent process ID: 1
This is an Orphan process now
```

10) Write a program

a) To make the process to sleep for a few seconds

CODE

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

int main() {
```

```
    pid_t pid;
    int n;
    pid=getpid();
    printf("\nThe Process ID is:%d",pid);

    printf("\nEnter the time in seconds for which you want to hibernate this process:");
    scanf("%d",&n);
    printf("\nThe Process will now sleep for %d second(s)",n);
    sleep(n);
    char str[3];
    sprintf(str,"%d",pid);
    char addr[]="cat /proc/";
    char ess[]="/status";
    strcat(addr,str);
    int r = system(addr);
    sleep(1);

    return 0;
}
```

OUTPUT

gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
int main() {

        pid_t pid;
        int n;
        pid=getpid();
        printf("\nThe Process ID is:%d",pid);

        printf("\nEnter the time in seconds for which you want to hibernate this process:");
        scanf("%d",&n);
        printf("\nThe Process will now sleep for %d second(s)",n);
        sleep(n);
        char str[3];
        sprintf(str,"%d",pid);
        char addr[]="cat /proc/";
        char ess[]="/status";
        strcat(addr,str);
        int r = system(addr);
        sleep(1);

        return 0;
}
~
```

gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2

```
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc Process_Sleep.c -o Process_Sleep
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./Process_Sleep

The Process ID is:145
Enter the time in seconds for which you want to hibernate this process:10
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc Process_Sleep.c -o Process_Sleep
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./Process_Sleep

The Process ID is:145
Enter the time in seconds for which you want to hibernate this process:10

Name:   Process_Sleep
State:  S (sleeping)
Tgid:   145
Pid:    145
PPid:   9
TracerPid:      0
Uid:    1000    1000    1000    1000
Gid:    1000    1000    1000    1000
FDSize: 3
Groups:
VmPeak: 0 kB
VmSize: 10536 kB
VmLck:  0 kB
VmHWM:  0 kB
VmRSS:  660 kB
VmData: 0 kB
VmStk:  0 kB
VmExe:  4 kB
VmLib:  0 kB
VmPTE:  0 kB
Threads:        1
SigQ:   0/0
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000000000000
SigCgt: 0000000000000000
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 0000001ffffffffff
Cpus_allowed:   ff
Cpus_allowed_list:      0-7
Mems_allowed:   1
Mems_allowed_list:      0
voluntary_ctxt_switches:        150
nonvoluntary_ctxt_switches:     545
*** stack smashing detected ***: terminated
Aborted (core dumped)
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
```

b) To create background process

CODE

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>
```

```
int main() {
	pid_t pid;
	pid=fork();
	if (pid==0){
		printf("\nPID of background process:%d",getpid());
		printf("\nBackground Process is now killed");
		sleep(10);
		exit(0);
	}
	else{
		printf("\nThis is Parent process working in foreground.\nChild Process is working in background for 10s.");
		printf("\nForeground Process Killed");
		sleep(1);
		kill(getpid());


	}
	return 0;
}
```

OUTPUT

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
int main() {
        pid_t pid;
        pid=fork();
        if (pid==0){
                printf("\nPID of background process:%d",getpid());
                printf("\nBackground Process is now killed");
                sleep(10);
                exit(0);
        }
        else{
                printf("\nThis is Parent process working in foreground.\nChild Process is working in background for 10s.");
                printf("\nForeground Process Killed");
                sleep(1);
                kill(getpid());

        }
        return 0;
}
~
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ vi Background_Process.c
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc Background_Process.c -o Background_Process
Background_Process.c: In function ‘main’:
Background_Process.c:19:3: warning: implicit declaration of function ‘kill’ [-Wimplicit-function-declaration]
   19 |   kill(getpid());
      |   ^~~~
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./Background_Process

This is Parent process working in foreground.

Child Process is working in background for 10s.
PID of background process:188
Foreground Process Killedgaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ps
  PID TTY          TIME CMD
    9 tty1     00:00:00 bash
  188 tty1     00:00:00 Background_Proc
  189 tty1     00:00:00 ps
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ Background Process is now killed
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ps
  PID TTY          TIME CMD
    9 tty1     00:00:00 bash
  190 tty1     00:00:00 ps
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
```

11) Implement the program to pass messages using pipes.

CODE

```
#include<stdio.h>

#include<unistd.h>


int main()

{

    int pipefds[2];

    int returnstatus;

    char writemessages[2][20]={"Hello", "World"};

    char readmessage[20];

    returnstatus = pipe(pipefds);

    if (returnstatus == -1) {

        printf("Unable to create pipe\n");

        return 1;

    }

    printf("Writing to pipe - Message 1 is %s\n", writemessages[0]);

    write(pipefds[1], writemessages[0], sizeof(writemessages[0]));
```

```
    read(pipefds[0], readmessage, sizeof(readmessage));
    printf("Reading from pipe – Message 1 is %s\n", readmessage);
    printf("Writing to pipe - Message 2 is %s\n", writemessages[0]);
    write(pipefds[1], writemessages[1], sizeof(writemessages[0]));
    read(pipefds[0], readmessage, sizeof(readmessage));
    printf("Reading from pipe – Message 2 is %s\n", readmessage);
    return 0;
}
```

OUTPUT

gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2

```
#include<stdio.h>
#include<unistd.h>

int main()
{
        int pipefds[2];
        int returnstatus;
        char writemessages[2][20]={"Hello", "World"};
        char readmessage[20];
        returnstatus = pipe(pipefds);
        if (returnstatus == -1) {
                printf("Unable to create pipe\n");
                return 1;
        }
        printf("Writing to pipe - Message 1 is %s\n", writemessages[0]);
        write(pipefds[1], writemessages[0], sizeof(writemessages[0]));
        read(pipefds[0], readmessage, sizeof(readmessage));
        printf("Reading from pipe – Message 1 is %s\n", readmessage);
        printf("Writing to pipe - Message 2 is %s\n", writemessages[0]);
        write(pipefds[1], writemessages[1], sizeof(writemessages[0]));
        read(pipefds[0], readmessage, sizeof(readmessage));
        printf("Reading from pipe – Message 2 is %s\n", readmessage);
        return 0;
}
~
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ vi Pipes.c
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc Pipes.c -o Pipes
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./Pipes
Writing to pipe - Message 1 is Hello
Reading from pipe – Message 1 is Hello
Writing to pipe - Message 2 is Hello
Reading from pipe – Message 2 is World
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
```

12) Write a program to demonstrate the implementation of Inter Process Communication (IPC) using shared memory.

CODE (Writer)

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/shm.h>

#include<string.h>

int main()

{

    int i;

    void *shared_memory;

    char buff[100];

    int shmid;

    shmid=shmget((key_t)2345, 1024, 0666|IPC_CREAT);

    printf("Key of shared memory is %d\n",shmid);

    shared_memory=shmat(shmid,NULL,0);

    printf("Process attached at %p\n",shared_memory);

    printf("Enter some data to write to shared memory\n");

    read(0,buff,100);

    strcpy(shared_memory,buff);

    printf("You wrote : %s\n",(char *)shared_memory);

}
```

OUTPUT (Writer)

gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
#include<string.h>
int main()
{
        int i;
        void *shared_memory;
        char buff[100];
        int shmid;
        shmid=shmget((key_t)2345, 1024, 0666|IPC_CREAT);
        printf("Key of shared memory is %d\n",shmid);
        shared_memory=shmat(shmid,NULL,0);
        printf("Process attached at %p\n",shared_memory);
        printf("Enter some data to write to shared memory\n");
        read(0,buff,100);
        strcpy(shared_memory,buff);
        printf("You wrote : %s\n",(char *)shared_memory);
}
~
```

gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2

```
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc IPCWriter.c -o IPCWriter
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./IPCWriter
Key of shared memory is 1
Process attached at 0x7f2bbe1fd000
Enter some data to write to shared memory
Hello World
You wrote : Hello World

gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
```

CODE (Reader)

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/shm.h>

#include<string.h>

int main()
```

```
{
    int i;
    void *shared_memory;
    char buff[100];
    int shmid;
    shmid=shmget((key_t)2345, 1024, 0666);
    printf("Key of shared memory is %d\n",shmid);
    shared_memory=shmat(shmid,NULL,0);
    printf("Process attached at %p\n",shared_memory);
    printf("Data read from shared memory is : %s\n",(char *)shared_memory);
}
```

OUTPUT (Reader)

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
#include<string.h>
int main()
{
        int i;
        void *shared_memory;
        char buff[100];
        int shmid;
        shmid=shmget((key_t)2345, 1024, 0666);
        printf("Key of shared memory is %d\n",shmid);
        shared_memory=shmat(shmid,NULL,0);
        printf("Process attached at %p\n",shared_memory);
        printf("Data read from shared memory is : %s\n",(char *)shared_memory);
}

~
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ vi IPCReader.c
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc IPCReader.c -o IPCReader
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./IPCReader
Key of shared memory is 1
Process attached at 0x7f99fb256000
Data read from shared memory is : Hello World

gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
```

13) Write a program to create a thread and let the thread check whether the given number is prime or not.

CODE

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

void *Prime(void* var){

	printf("\nThis is thread process with pid:%d",getpid());

	int flag=0;

	int n=(int *)var;

	for(int i=2;i<=n/2;i++){

		if(n%i==0){

			flag =1;

			break;

		}

	}

	if (n==1) {

		printf("\n1 is neither prime nor composite");

	}

	else {

		if (flag==0) {

			printf("\n%d is Prime.\n",n);

		}

		else {

			printf("\n%d is Composite.\n",n);

		}

	}

}

int main() {

	int n;
```

```
    printf("\nThis is parent process with pid:%d",getpid());
    printf("\nBefore Thread Creation.\nThread will now be Created...");
    printf("\nEnter the number which u want to check for prime: ");
    scanf("%d",&n);
    pthread_t tid;
    pthread_create (&tid,NULL,Prime,(void*)n);
    pthread_join(tid,NULL);
    pthread_exit(NULL);
    exit(0);
}
```

OUTPUT

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

void *Prime(void* var){
        printf("\nThis is thread process with pid:%d",getpid());
        int flag=0;
        int n=(int *)var;
        for(int i=2;i<=n/2;i++){
                if(n%i==0){
                        flag =1;
                        break;
                }
        }
        if (n==1) {
                printf("\n1 is neither prime nor composite");
        }
        else {
                if (flag==0) {
                        printf("\n%d is Prime.\n",n);
                }
                else {
                        printf("\n%d is Composite.\n",n);
                }
        }

}


int main() {
        int n;
        printf("\nThis is parent process with pid:%d",getpid());
        printf("\nBefore Thread Creation.\nThread will now be Created...");
        printf("\nEnter the number which u want to check for prime: ");
        scanf("%d",&n);
        pthread_t tid;


        pthread_create (&tid,NULL,Prime,n);
        pthread_join(tid,NULL);
        pthread_exit(NULL);
        exit(0);
}
~
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ vi Threading.c
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc Threading.c -o Threading -lpthread
Threading.c: In function ‘Prime’:
Threading.c:9:8: warning: initialization of ‘int’ from ‘int *’ makes integer from pointer without a cast [-Wint-conversion]
    9 |  int n=(int *)var;
      |        ^
Threading.c: In function ‘main’:
Threading.c:40:34: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
   40 |  pthread_create (&tid,NULL,Prime,(void*)n);
      |                                  ^
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./Threading

This is parent process with pid:331
Before Thread Creation.
Thread will now be Created...
Enter the number which u want to check for prime: 34

This is thread process with pid:331
34 is Composite.
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./Threading

This is parent process with pid:333
Before Thread Creation.
Thread will now be Created...
Enter the number which u want to check for prime: 23

This is thread process with pid:333
23 is Prime.
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
```

14) Design the following CPU Scheduling Algorithms to provide the performance analysis among them.

a) FCFS

CODE

```
#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <stdlib.h>

#include <string.h>


struct Process {

    int p_id;

    int AT;

    int BT;

    int CT;

    int TAT;

    int WT;
```

```
};

void display(struct Process Process_Array[], int n){
	for (int i=0; i<n;i++) {
		int Pno=i+1;
		printf("\n\nProcess %d\n",Pno);
		printf("p_id=%d\n",Process_Array[i].p_id);
		printf("AT=%d\n",Process_Array[i].AT);
		printf("BT=%d\n",Process_Array[i].BT);
		printf("CT=%d\n",Process_Array[i].CT);
		printf("TAT=%d\n",Process_Array[i].TAT);
		printf("WT=%d\n",Process_Array[i].WT);
	}
}
void getStats(struct Process Process_Array[], int n){
	printf("\nEnter the details of every process in increasing order of their arrival times.\n");
	for (int i=0; i<n;i++) {
		printf("Enter PID of Process %d = ",i+1);
		scanf("%d",&Process_Array[i].p_id);
		printf("Enter Arrival Time of Process %d = ",i+1);
		scanf("%d",&Process_Array[i].AT);
		printf("Enter Burst Time of Process %d = ",i+1);
		scanf("%d",&Process_Array[i].BT);
	}
}
void calcCT(struct Process Process_Array[],int n){
	int timeline=0;
	for (int i=0; i<n;i++) {
		if (timeline<Process_Array[i].AT) {
			timeline=Process_Array[i].AT;
		}
```

```
        timeline = timeline + Process_Array[i].BT;
        Process_Array[i].CT = timeline;
    }
}
void calcTAT(struct Process Process_Array[],int n){
    for (int i=0; i<n;i++) {
        Process_Array[i].TAT = Process_Array[i].CT - Process_Array[i].AT;
    }
}
void calcWT(struct Process Process_Array[],int n){
    for (int i=0; i<n;i++) {
        Process_Array[i].WT = Process_Array[i].TAT - Process_Array[i].BT;
    }
}
void calcAvgTAT(struct Process Process_Array[],int n){
    float sumTAT=0;
    for (int i=0; i<n;i++) {
        sumTAT = sumTAT + Process_Array[i].TAT;
    }
    printf("\n\nAverage Turnaround time= %.3f", (sumTAT/n));
}
void calcAvgWT(struct Process Process_Array[],int n){
    float sumWT=0;
    for (int i=0; i<n;i++) {
        sumWT = sumWT + Process_Array[i].WT;
    }
    printf("\n\nAverage Waiting time= %.3f\n", (sumWT/n));
}
int main(){
    int n;
    printf("\nEnter the number of processes in the system:");
```

```
    scanf ("%d",&n);
    struct Process Process_Array[100];
    getStats(Process_Array, n);
    calcCT(Process_Array, n);
    calcTAT(Process_Array, n);
    calcWT(Process_Array, n);
    display(Process_Array, n);
    calcAvgTAT(Process_Array, n);
    calcAvgWT(Process_Array, n);
    return 0;
}
```

OUTPUT

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
        int BT;
        int CT;
        int TAT;
        int WT;
};

void display(struct Process Process_Array[], int n){
        for (int i=0; i<n;i++) {
                int Pno=i+1;
                printf("\n\nProcess %d\n",Pno);
                printf("p_id=%d\n",Process_Array[i].p_id);
                printf("AT=%d\n",Process_Array[i].AT);
                printf("BT=%d\n",Process_Array[i].BT);
                printf("CT=%d\n",Process_Array[i].CT);
                printf("TAT=%d\n",Process_Array[i].TAT);
                printf("WT=%d\n",Process_Array[i].WT);
        }
}
void getStats(struct Process Process_Array[], int n){
        printf("\nEnter the details of every process in increasing order of their arrival times.\n");
        for (int i=0; i<n;i++) {
                printf("Enter PID of Process %d = ",i+1);
                scanf("%d",&Process_Array[i].p_id);
                printf("Enter Arrival Time of Process %d = ",i+1);
                scanf("%d",&Process_Array[i].AT);
                printf("Enter Burst Time of Process %d = ",i+1);
                scanf("%d",&Process_Array[i].BT);
        }
}
void calcCT(struct Process Process_Array[],int n){
        int timeline=0;
        for (int i=0; i<n;i++) {
                if (timeline<Process_Array[i].AT) {
                        timeline=Process_Array[i].AT;
                }
                timeline = timeline + Process_Array[i].BT;
                Process_Array[i].CT = timeline;
        }
}
void calcTAT(struct Process Process_Array[],int n){
        for (int i=0; i<n;i++) {
                Process_Array[i].TAT = Process_Array[i].CT - Process_Array[i].AT;
        }
}
void calcWT(struct Process Process_Array[],int n){
        for (int i=0; i<n;i++) {
                Process_Array[i].WT = Process_Array[i].TAT - Process_Array[i].BT;
        }
}
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc FCFS.c -o FCFS
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./FCFS

Enter the number of processes in the system:5

Enter the details of every process in increasing order of their arrival times.
Enter PID of Process 1 = 1
Enter Arrival Time of Process 1 = 0
Enter Burst Time of Process 1 = 12
Enter PID of Process 2 = 2
Enter Arrival Time of Process 2 = 1
Enter Burst Time of Process 2 = 8
Enter PID of Process 3 = 3
Enter Arrival Time of Process 3 = 2
Enter Burst Time of Process 3 = 7
Enter PID of Process 4 = 4
Enter Arrival Time of Process 4 = 7
Enter Burst Time of Process 4 = 4
Enter PID of Process 5 = 5
Enter Arrival Time of Process 5 = 8
Enter Burst Time of Process 5 = 6


Process 1
p_id=1
AT=0
BT=12
CT=12
TAT=12
WT=0


Process 2
p_id=2
AT=1
BT=8
CT=20
TAT=19
WT=11


Process 3
p_id=3
AT=2
BT=7
CT=27
TAT=25
WT=18
```

gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2

```
Process 1
p_id=1
AT=0
BT=12
CT=12
TAT=12
WT=0

Process 2
p_id=2
AT=1
BT=8
CT=20
TAT=19
WT=11

Process 3
p_id=3
AT=2
BT=7
CT=27
TAT=25
WT=18

Process 4
p_id=4
AT=7
BT=4
CT=31
TAT=24
WT=20

Process 5
p_id=5
AT=8
BT=6
CT=37
TAT=29
WT=23

Average Turnaround time= 21.800

Average Waiting time= 14.400
```

b) PRIORITY

CODE

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <stdlib.h>
#include <string.h>

struct Process {
	int p_id;
	int AT;
	int BT;
	int rem_BT;
	int CT;
	int TAT;
	int WT;
	int Priority;
};

void display(struct Process Process_Array[], int n){
	for (int i=0; i<n;i++) {
		int Pno=i+1;
		printf("\n\nProcess %d\n",Pno);
		printf("p_id=%d\n",Process_Array[i].p_id);
		printf("AT=%d\n",Process_Array[i].AT);
		printf("BT=%d\n",Process_Array[i].BT);
		printf("Priority=%d\n",Process_Array[i].Priority);
		printf("CT=%d\n",Process_Array[i].CT);
		printf("TAT=%d\n",Process_Array[i].TAT);
		printf("WT=%d\n",Process_Array[i].WT);
	}
```

```
}
void getStats(struct Process Process_Array[], int n){
	printf("\nEnter the details of every process in increasing order of their arrival times.\n");
	for (int i=0; i<n;i++) {
		printf("Enter PID of Process %d = ",i+1);
		scanf("%d",&Process_Array[i].p_id);
		printf("Enter Arrival Time of Process %d = ",i+1);
		scanf("%d",&Process_Array[i].AT);
		printf("Enter Burst Time of Process %d = ",i+1);
		scanf("%d",&Process_Array[i].BT);
		Process_Array[i].rem_BT=Process_Array[i].BT;
		printf("Enter Priority of Process %d = ",i+1);
		scanf("%d",&Process_Array[i].Priority);
	}
}
int readyQueueManagement(struct Process Process_Array[], struct Process Ready[],int timeline, int
n){
	int i=0, j=0;
	while (Process_Array[i].AT<=timeline && i<n){
		if (Process_Array[i].rem_BT !=0){
			Ready[j]=Process_Array[i];
			j++;
		}
		i++;
	}
	return j;
}


int MaxPriority(struct Process Ready[], int j){
	int max=0;
	for(int i=0; i<j;i++) {
```

```
            if (Ready[i].Priority>Ready[max].Priority){
                max=i;
            }
        }
        return max;
}

void calcCT(struct Process Process_Array[], struct Process Ready[],int n){
        int timeline=0;
        int counter = 0;
        while (counter !=n) {
            int j=readyQueueManagement(Process_Array, Ready, timeline, n);
            if (j==0){
                timeline++;
            }
            else{
                int max= MaxPriority(Ready, j);
                for(int i=0;i<n;i++) {
                    if (Ready[max].p_id==Process_Array[i].p_id){
                        Process_Array[i].rem_BT=Process_Array[i].rem_BT-1;
                        timeline=timeline+1;
                        if (Process_Array[i].rem_BT==0){
                            Process_Array[i].CT=timeline;
                            counter++;
                        }
                    }
            }
        }
                                                    }
}
void calcTAT(struct Process Process_Array[],int n){
```

```
	for (int i=0; i<n;i++) {
		Process_Array[i].TAT = Process_Array[i].CT - Process_Array[i].AT;
	}
}
void calcWT(struct Process Process_Array[],int n){
	for (int i=0; i<n;i++) {
		Process_Array[i].WT = Process_Array[i].TAT - Process_Array[i].BT;
	}
}
void calcAvgTAT(struct Process Process_Array[],int n){
	float sumTAT=0;
	for (int i=0; i<n;i++) {
		sumTAT = sumTAT + Process_Array[i].TAT;
	}
	printf("\n\nAverage Turnaround time= %.3f", (sumTAT/n));
}
void calcAvgWT(struct Process Process_Array[],int n){
	float sumWT=0;
	for (int i=0; i<n;i++) {
		sumWT = sumWT + Process_Array[i].WT;
	}
	printf("\n\nAverage Waiting time= %.3f", (sumWT/n));
}
int main(){
	int n;
	printf("\nEnter the number of Processes in the system:");
	scanf ("%d",&n);
	struct Process Process_Array[100];
	struct Process Ready[100];
	getStats(Process_Array, n);
	calcCT(Process_Array,Ready, n);
```

```
    calcTAT(Process_Array, n);
    calcWT(Process_Array, n);
    display(Process_Array, n);
    calcAvgTAT(Process_Array, n);
    calcAvgWT(Process_Array, n);
    return 0;
}
```

OUTPUT

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
                Process_Array[i].rem_BT=Process_Array[i].BT;
                printf("Enter Priority of Process %d = ",i+1);
                scanf("%d",&Process_Array[i].Priority);
        }
}

int readyQueueManagement(struct Process Process_Array[], struct Process Ready[],int timeline, int n){
        int i=0, j=0;
        while (Process_Array[i].AT<=timeline && i<n){
                if (Process_Array[i].rem_BT !=0){
                        Ready[j]=Process_Array[i];
                        j++;
                }
                i++;
        }
        return j;
}

int MaxPriority(struct Process Ready[], int j){
        int max=0;
        for(int i=0; i<j;i++) {
                if (Ready[i].Priority>Ready[max].Priority){
                        max=i;
                }
        }
        return max;
}

void calcCT(struct Process Process_Array[], struct Process Ready[],int n){
        int timeline=0;
        int counter = 0;
        while (counter !=n) {
                int j=readyQueueManagement(Process_Array, Ready, timeline, n);
                if (j==0){
                        timeline++;
                }
                else{
                        int max= MaxPriority(Ready, j);
                        for(int i=0;i<n;i++) {
                                if (Ready[max].p_id==Process_Array[i].p_id){
                                        Process_Array[i].rem_BT=Process_Array[i].rem_BT-1;
                                        timeline=timeline+1;
                                        if (Process_Array[i].rem_BT==0){
                                                Process_Array[i].CT=timeline;
                                                counter++;
                                        }
                                }
                        }
                }
        }
                                                                                        }
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc Priority_Scheduler.c -o Priority_Scheduling
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./Priority_Scheduling

Enter the number of Processes in the system:4

Enter the details of every process in increasing order of their arrival times.
Enter PID of Process 1 = 1
Enter Arrival Time of Process 1 = 0
Enter Burst Time of Process 1 = 5
Enter Priority of Process 1 = 10
Enter PID of Process 2 = 2
Enter Arrival Time of Process 2 = 1
Enter Burst Time of Process 2 = 4
Enter Priority of Process 2 = 20
Enter PID of Process 3 = 3
Enter Arrival Time of Process 3 = 2
Enter Burst Time of Process 3 = 2
Enter Priority of Process 3 = 30
Enter PID of Process 4 = 4
Enter Arrival Time of Process 4 = 4
Enter Burst Time of Process 4 = 1
Enter Priority of Process 4 = 40


Process 1
p_id=1
AT=0
BT=5
Priority=10
CT=12
TAT=12
WT=7


Process 2
p_id=2
AT=1
BT=4
Priority=20
CT=8
TAT=7
WT=3
```

```
Process 1
p_id=1
AT=0
BT=5
Priority=10
CT=12
TAT=12
WT=7


Process 2
p_id=2
AT=1
BT=4
Priority=20
CT=8
TAT=7
WT=3


Process 3
p_id=3
AT=2
BT=2
Priority=30
CT=4
TAT=2
WT=0


Process 4
p_id=4
AT=4
BT=1
Priority=40
CT=5
TAT=1
WT=0


Average Turnaround time= 5.500

Average Waiting time= 2.500gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
```

c) ROUND ROBIN

CODE

```
#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <stdlib.h>

#include <string.h>


struct Process {

	int p_id;

	int AT;

	int BT;

	int rem_BT;

	int CT;

	int TAT;

	int WT;

}buffer;


void SeT(struct Process Ready[], int n){

	buffer.p_id=-1;

	for(int i=0;i<n;i++){

		Ready[i].p_id=-1;

		Ready[i].AT=-1;

		Ready[i].BT=-1;

		Ready[i].rem_BT=-1;

		Ready[i].CT=-1;

		Ready[i].TAT=-1;

		Ready[i].WT=-1;

	}

}
```

```
void display(struct Process Process_Array[], int n){
	for (int i=0; i<n;i++) {
		int Pno=i+1;
		printf("\n\nProcess %d\n",Process_Array[i].p_id);
		printf("p_id=%d\n",Process_Array[i].p_id);
		printf("AT=%d\n",Process_Array[i].AT);
		printf("BT=%d\n",Process_Array[i].BT);
		printf("CT=%d\n",Process_Array[i].CT);
		printf("TAT=%d\n",Process_Array[i].TAT);
		printf("WT=%d\n",Process_Array[i].WT);
		printf("Rem_BT=%d\n",Process_Array[i].rem_BT);
	}
}
void getStats(struct Process Process_Array[], int n){
	printf("\nEnter the details of every process in increasing order of their arrival times.\n");
	for (int i=0; i<n;i++) {
		printf("Enter PID of Process %d = ",i+1);
		scanf("%d",&Process_Array[i].p_id);
		printf("Enter Arrival Time of Process %d = ",i+1);
		scanf("%d",&Process_Array[i].AT);
		printf("Enter Burst Time of Process %d = ",i+1);
		scanf("%d",&Process_Array[i].BT);
		Process_Array[i].rem_BT=Process_Array[i].BT;
	}
}

void queue(struct Process Ready[], struct Process Process_Array){
	int i=0;
	while(Ready[i].p_id!=-1){
		i++;
	}
```

```
    Ready[i]=Process_Array;
}



struct Process dequeue(struct Process Ready[]){
    struct Process temp=Ready[0];
    int i=0;
    while(Ready[i].p_id!=-1){
        i++;
    }
    int j=0;
    for (j; j<i+1;j++){
        Ready[j]=Ready[j+1];
    }
    return temp;
}

int Ready_No(struct Process Ready[]){
    int i=0;
    while(Ready[i].p_id!=-1){
        i++;
    }
    return i;
}
void calcCT(struct Process Process_Array[], struct Process Ready[], int Time_Quantum, int n){
    int timeline=0;
    int counter=0;
    int Process_counter=0;
    while(counter !=n){
        for(int i=Process_counter;i<n;i++){
            if(Process_Array[i].AT<=timeline){
```

```
                queue(Ready, Process_Array[i]);
                Process_counter++;
            }
        }
        int ReadyQueueCheck=Ready_No(Ready);
        if(ReadyQueueCheck==0){
            timeline++;
        }
        else{
            struct Process temp=dequeue(Ready);
            if(temp.rem_BT<=Time_Quantum){
                for(int i=0;i<n;i++){
                    if(Process_Array[i].p_id==temp.p_id){
                        timeline=timeline+Process_Array[i].rem_BT;
                        Process_Array[i].rem_BT=0;
                        Process_Array[i].CT=timeline;
                        counter++;
                    }
                }
            }
            else{
                for(int i=0;i<n;i++){
                    if(Process_Array[i].p_id==temp.p_id){
                        timeline=timeline+Time_Quantum;
                        for(int i=Process_counter;i<n;i++){
                            if(Process_Array[i].AT<=timeline){
                                queue(Ready, Process_Array[i]);
                                Process_counter++;
                            }
                        }
                    Process_Array[i].rem_BT=Process_Array[i].rem_BT-Time_Quantum;
```

```
                    queue(Ready, Process_Array[i]);
                    }
                }
            }
        }
    }
}
void calcTAT(struct Process Process_Array[],int n){
    for (int i=0; i<n;i++) {
        Process_Array[i].TAT = Process_Array[i].CT - Process_Array[i].AT;
    }
}
void calcWT(struct Process Process_Array[],int n){
    for (int i=0; i<n;i++) {
        Process_Array[i].WT = Process_Array[i].TAT - Process_Array[i].BT;
    }
}
void calcAvgTAT(struct Process Process_Array[],int n){
    float sumTAT=0;
    for (int i=0; i<n;i++) {
        sumTAT = sumTAT + Process_Array[i].TAT;
    }
    printf("\n\nAverage Turnaround time= %.3f", (sumTAT/n));
}
void calcAvgWT(struct Process Process_Array[],int n){
    float sumWT=0;
    for (int i=0; i<n;i++) {
        sumWT = sumWT + Process_Array[i].WT;
    }
    printf("\n\nAverage Waiting time= %.3f", (sumWT/n));
}
```

```
int main(){
    int n, Time_Quantum;
    printf("\nEnter Number of Processes in the system: ");
    scanf ("%d",&n);
    printf("Enter Time Quantum: ");
    scanf("%d",&Time_Quantum);
    struct Process Process_Array[100];
    struct Process Ready[100];
    SeT(Ready, 100);
    struct Process temp=dequeue(Ready);
    getStats(Process_Array, n);
    calcCT(Process_Array,Ready, Time_Quantum, n);
    calcTAT(Process_Array, n);
    calcWT(Process_Array, n);
    display(Process_Array, n);
    calcAvgTAT(Process_Array, n);
    calcAvgWT(Process_Array, n);
    return 0;
}
```

OUTPUT

gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2

```
void calcCT(struct Process Process_Array[], struct Process Ready[], int Time_Quantum, int n){
        int timeline=0;
        int counter=0;
        int Process_counter=0;
        while(counter !=n){
                for(int i=Process_counter;i<n;i++){
                        if(Process_Array[i].AT<=timeline){
                                queue(Ready, Process_Array[i]);
                                Process_counter++;
                        }
                }
                int ReadyQueueCheck=Ready_No(Ready);
                if(ReadyQueueCheck==0){
                        timeline++;
                }
                else{
                        struct Process temp=dequeue(Ready);
                        if(temp.rem_BT<=Time_Quantum){
                                for(int i=0;i<n;i++){
                                        if(Process_Array[i].p_id==temp.p_id){
                                                timeline=timeline+Process_Array[i].rem_BT;
                                                Process_Array[i].rem_BT=0;
                                                Process_Array[i].CT=timeline;
                                                counter++;
                                        }
                                }
                        }
                        else{
                                for(int i=0;i<n;i++){
                                        if(Process_Array[i].p_id==temp.p_id){
                                                timeline=timeline+Time_Quantum;
                                                for(int i=Process_counter;i<n;i++){
                                                        if(Process_Array[i].AT<=timeline){
                                                                queue(Ready, Process_Array[i]);
                                                                Process_counter++;
                                                        }
                                                }
                                        Process_Array[i].rem_BT=Process_Array[i].rem_BT-Time_Quantum;
                                        queue(Ready, Process_Array[i]);
                                        }
                                }
                        }
                }
        }
    }
}
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc Round_Robin.c -o Round_Robin
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./Round_Robin

Enter Number of Processes in the system: 5
Enter Time Quantum: 6

Enter the details of every process in increasing order of their arrival times.
Enter PID of Process 1 = 1
Enter Arrival Time of Process 1 = 0
Enter Burst Time of Process 1 = 12
Enter PID of Process 2 = 2
Enter Arrival Time of Process 2 = 1
Enter Burst Time of Process 2 = 8
Enter PID of Process 3 = 3
Enter Arrival Time of Process 3 = 2
Enter Burst Time of Process 3 = 7
Enter PID of Process 4 = 4
Enter Arrival Time of Process 4 = 7
Enter Burst Time of Process 4 = 4
Enter PID of Process 5 = 5
Enter Arrival Time of Process 5 = 8
Enter Burst Time of Process 5 = 6


Process 1
p_id=1
AT=0
BT=12
CT=24
TAT=24
WT=12
Rem_BT=0


Process 2
p_id=2
AT=1
BT=8
CT=36
TAT=35
WT=27
Rem_BT=0
```

```
Process 2
p_id=2
AT=1
BT=8
CT=36
TAT=35
WT=27
Rem_BT=0


Process 3
p_id=3
AT=2
BT=7
CT=37
TAT=35
WT=28
Rem_BT=0


Process 4
p_id=4
AT=7
BT=4
CT=28
TAT=21
WT=17
Rem_BT=0


Process 5
p_id=5
AT=8
BT=6
CT=34
TAT=26
WT=20
Rem_BT=0


Average Turnaround time= 28.200

Average Waiting time= 20.800gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
```

d) SJF

CODE

```
#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <stdlib.h>

#include <string.h>
```

```
struct Process {
	int p_id;
	int AT;
	int BT;
	int CT;
	int TAT;
	int WT;
	int flag;
};

void display(struct Process Process_Array[], int n){

	for (int i=0; i<n;i++) {
		int Pno=i+1;
		printf("\n\nProcess %d\n",Pno);
		printf("p_id=%d\n",Process_Array[i].p_id);
		printf("AT=%d\n",Process_Array[i].AT);
		printf("BT=%d\n",Process_Array[i].BT);
		printf("CT=%d\n",Process_Array[i].CT);
		printf("TAT=%d\n",Process_Array[i].TAT);
		printf("WT=%d\n",Process_Array[i].WT);
	}
}
void getStats(struct Process Process_Array[], int n){
	printf("\nEnter the details of every process in increasing order of their arrival times.\n");
	for (int i=0; i<n;i++) {
		printf("Enter PID of Process %d = ",i+1);
		scanf("%d",&Process_Array[i].p_id);
		printf("Enter Arrival Time of Process %d = ",i+1);
		scanf("%d",&Process_Array[i].AT);
		printf("Enter Burst Time of Process %d = ",i+1);
```

```
            scanf("%d",&Process_Array[i].BT);
            Process_Array[i].flag=0;
        }
}
int MinBT(struct Process Process_Array[],int n){
    int min=0;
    for (int i=0; i<n; i++){
        if (Process_Array[i].BT<Process_Array[min].BT && Process_Array[i].flag==0){
            min=i;
        }
    }
    return min;
}



void calcCT(struct Process Process_Array[], struct Process Ready[], int n){
        int timeline=0;
        int counter=0;
        int min=0;
        int j=0;
        while (counter!=n){
            j=0;
            for (int i=0; i<n; i++){
                if(Process_Array[i].AT<=timeline&& Process_Array[i].flag==0){
                    Ready[j]=Process_Array[i];
                    j++;
                }
            }
            if (j==0){
                timeline++;
            }
```

```
        else{
            min=MinBT(Ready,j);
            for(int i=0; i<n; i++){
                if (Process_Array[i].p_id==Ready[min].p_id){
                    timeline=timeline+Process_Array[i].BT;
                    Process_Array[i].CT=timeline;
                    Process_Array[i].flag=1;
                    counter++;
                }
            }
        }
    }
}
void calcTAT(struct Process Process_Array[],int n){
    for (int i=0; i<n;i++) {
        Process_Array[i].TAT = Process_Array[i].CT - Process_Array[i].AT;
    }
}
void calcWT(struct Process Process_Array[],int n){
    for (int i=0; i<n;i++) {
        Process_Array[i].WT = Process_Array[i].TAT - Process_Array[i].BT;
    }
}
void calcAvgTAT(struct Process Process_Array[],int n){
    float sumTAT=0;
    for (int i=0; i<n;i++) {
        sumTAT = sumTAT + Process_Array[i].TAT;
    }
    printf("\n\nAverage Turnaround time= %.3f", (sumTAT/n));
}
void calcAvgWT(struct Process Process_Array[],int n){
```

```
    float sumWT=0;
    for (int i=0; i<n;i++) {
        sumWT = sumWT + Process_Array[i].WT;
    }
    printf("\n\nAverage Waiting time= %.3f", (sumWT/n));
}
int main(){
    int n;
    printf("\nEnter the number of processes in the system:");
    scanf ("%d",&n);
    struct Process Process_Array[100];
    struct Process Ready[100];
    getStats(Process_Array, n);
    calcCT(Process_Array, Ready, n);
    calcTAT(Process_Array, n);
    calcWT(Process_Array, n);
    display(Process_Array, n);
    calcAvgTAT(Process_Array, n);
    calcAvgWT(Process_Array, n);
    return 0;
}
```

OUTPUT

gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2

```
}
int MinBT(struct Process Process_Array[],int n){
        int min=0;
        for (int i=0; i<n; i++){
                if (Process_Array[i].BT<Process_Array[min].BT && Process_Array[i].flag==0){
                        min=i;
                }
        }
        return min;
}


void calcCT(struct Process Process_Array[], struct Process Ready[], int n){
        int timeline=0;
        int counter=0;
        int min=0;
        int j=0;
        while (counter!=n){
                j=0;
                for (int i=0; i<n; i++){
                        if(Process_Array[i].AT<=timeline&& Process_Array[i].flag==0){
                                Ready[j]=Process_Array[i];
                                j++;
                        }
                }
                if (j==0){
                        timeline++;
                }
                else{
                        min=MinBT(Ready,j);
                        for(int i=0; i<n; i++){
                                if (Process_Array[i].p_id==Ready[min].p_id){
                                        timeline=timeline+Process_Array[i].BT;
                                        Process_Array[i].CT=timeline;
                                        Process_Array[i].flag=1;
                                        counter++;
                                }
                        }
                }
        }
}
void calcTAT(struct Process Process_Array[],int n){
        for (int i=0; i<n;i++) {
                Process_Array[i].TAT = Process_Array[i].CT - Process_Array[i].AT;
        }
}
void calcWT(struct Process Process_Array[],int n){
        for (int i=0; i<n;i++) {
                Process_Array[i].WT = Process_Array[i].TAT - Process_Array[i].BT;
        }
}
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ gcc SJF.c -o SJF
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$ ./SJF

Enter the number of processes in the system 4

Enter the details of every process in increasing order of their arrival times.
Enter PID of Process 1 = 1
Enter Arrival Time of Process 1 = 1
Enter Burst Time of Process 1 = 3
Enter PID of Process 2 = 2
Enter Arrival Time of Process 2 = 2
Enter Burst Time of Process 2 = 4
Enter PID of Process 3 = 3
Enter Arrival Time of Process 3 = 1
Enter Burst Time of Process 3 = 2
Enter PID of Process 4 = 4
Enter Arrival Time of Process 4 = 4
Enter Burst Time of Process 4 = 4


Process 1
p_id=1
AT=1
BT=3
CT=6
TAT=5
WT=2


Process 2
p_id=2
AT=2
BT=4
CT=10
TAT=8
WT=4


Process 3
p_id=3
AT=1
BT=2
CT=3
TAT=2
WT=0
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet2
Enter PID of Process 2 = 2
Enter Arrival Time of Process 2 = 2
Enter Burst Time of Process 2 = 4
Enter PID of Process 3 = 3
Enter Arrival Time of Process 3 = 1
Enter Burst Time of Process 3 = 2
Enter PID of Process 4 = 4
Enter Arrival Time of Process 4 = 4
Enter Burst Time of Process 4 = 4


Process 1
p_id=1
AT=1
BT=3
CT=6
TAT=5
WT=2


Process 2
p_id=2
AT=2
BT=4
CT=10
TAT=8
WT=4


Process 3
p_id=3
AT=1
BT=2
CT=3
TAT=2
WT=0


Process 4
p_id=4
AT=4
BT=4
CT=14
TAT=10
WT=6


Average Turnaround time= 6.250

Average Waiting time= 3.000gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet2$
```