Task 4. [CO1]: Interpret I/O System calls of UNIX operating system (open, read, write)

A) Execute the program in C that creates a child process, waits for the termination of the child and lists its PID, together with the state in which the process was terminated (in decimal and hexadecimal).

Aim

To Write a C program to create a child process and allow the parent to display "parent" and the child to display "child" on the screen

ALGORITHM:

STEP 1: Start the program.
STEP 2: Create child process using fork.
STEP 3: If the value of variable is equal to zero print child process and parents wait for child to finish then parent process will be executed.
STEP 4: Create child to check for error in fork.
STEP 5: Stop the program

Program

```
#include <stdio.h>
#include <sys/wait.h> /* contains prototype for wait */
int main(void)
{
int pid;
int status;
printf("Hello World!\n");
pid = fork( );
if(pid == -1) /* check for error in fork */
{
perror("bad fork");
exit(1);
}
if (pid == 0)
printf("I am the child process.\n");
else
{
wait(&status); /* parent waits for child to finish */
printf("I am the parent process.\n");
}
}
```

Output:

Hello World!
I am the child process.
I am the parent process

B) Consider a process P1 that forks P2, P2 forks P3, and P3 forks P4. P1 and P2 continue to execute while P3 terminates. Now, when P4 terminates, which process must wait for and reap P4?

Aim:

To implement in C for the following UNIX commands using System calls : cat and mv

ALGORITHM:

STEP 1: Start the program.
STEP 2: Create process using System call for cat and copy from one file to another file.
STEP 3: If the file will be open, add the content in the file.
STEP 4: otherwise file will be read only, display message as file open error.
STEP 5: Stop the program

Program:

```
 #include<sys/types.h>
#include<sys/stat.h>
#include<stdio.h>
#include<fcntl.h>
main( int argc,char *argv[3] )
{
int fd,i;
char buf[2];
fd=open(argv[1],O_RDONLY,0777);
if(fd==-argc)
{
printf("file open error");
}
else
{
while((i=read(fd,buf,1))>0)
{
printf("%c",buf[0]);
}
close(fd);
}
}
```

Output:

```
student@ubuntu:~$gcc –o prgcat.out prgcat.c
student@ubuntu:~$cat > ff
hello
hai
student@ubuntu:~$./prgcat.out ff
hello
hai
```