University of Engineering and Technology, Lahore

Department of Computer

Science

Operating Systems Lab

Lab # 3

Operating System

(Fork, getpid, getppid, wait)

- Make a word document with the convention "SECTION_ROLLNO _LAB-NO". In addition, paste all of your work done at the LINUX prompt.
- You have to submit a Word File.
- Plagiarism is strictly prohibited; negative marks would be given to students who cheat.

Task 01:

Create a child process using the fork system call then print the process id and parent process id of all running processes.

Task 02:

Execute the following loop in your program

```
for (int i = 0; i < 3; i++)
{
	fork();
}
cout << "Hello from the process " << getpid() << endl;
```

predict the output of the program,
is it same as your predicted output?

Task 03 Snippets:

```
int rank = 0;

for(int i = 1; i <= 2; ++i)
{
	if (fork() == 0)
	{
```

```
                rank = rank + i;
                break;
            }
        }
```

Task 03:

Write a program that launches four processes using fork system call

Process 0 display the number between 1 and 25
Process 1 displays the numbers between 26 and 50
Process 2 displays the numbers between 51 and 75
Process 3 displays the numbers between 76 and 100

Task 04:

Write a program that launches four processes using fork system call, then all the processes counts that how many prime numbers exists between 2 and 100,001.

Now process 0 should find the count between --- 2 to 25,001
process 1 should find the count between --- 25,002 to 50,001
process 2 should find the count between --- 50,002 to 75,001
process 3 should find the count between --- 75,002 to 100,001

Below there a function is given which finds out whether a number if prime or not

```
bool isPrime(int num)
{
	if (num == 1)
	{
		return true;
	}
	else
	{
		double result;
		int divisor = num - 1;
		while (num != -1)
		{
			result = num % divisor;
			if (result == 0)
			{
				num = -1;
			}
			else
			{
				divisor = divisor - 1;
			}
		}
```

```
                if (divisor == 1)
                {
                        return true;
                }
                else
                {
                        return false;
                }
        }
}
```

Task 5

Write a C/C++ Program to to create a great grandchild of a process i.e.,

Parent-> Child -> Child -> Child,

Do incorporate checks to ensure that no extra siblings

are created.

Task 6

AIM:

To load an executable program in a child processes exec system call.

ALGORITHM:

1. Firstly create a text file named data.txt in which write “My name is _____ (YOUR NAME) and my roll number is _____(YOUR ROLL NUMBER)“.

2. Create a program named “test.c”. In this program read this text file.

3. Now create a new program named “exec.c”. In this program create a child process and in that child load the program “test.c”.

4. If return value is -1 then

- **a.** Print "Process creation unsuccessful"
- **b.** Terminate using exit system call.

5. Stop.

Hints:

```
#include <unistd.h> /* for fork */

#include <sys/types.h> /* for pid_t */

#include <sys/wait.h> /* for wait */
```