**Task 1**: Write what you have learned in few lines on each of the three programs that were using the ***fork()*** system call.

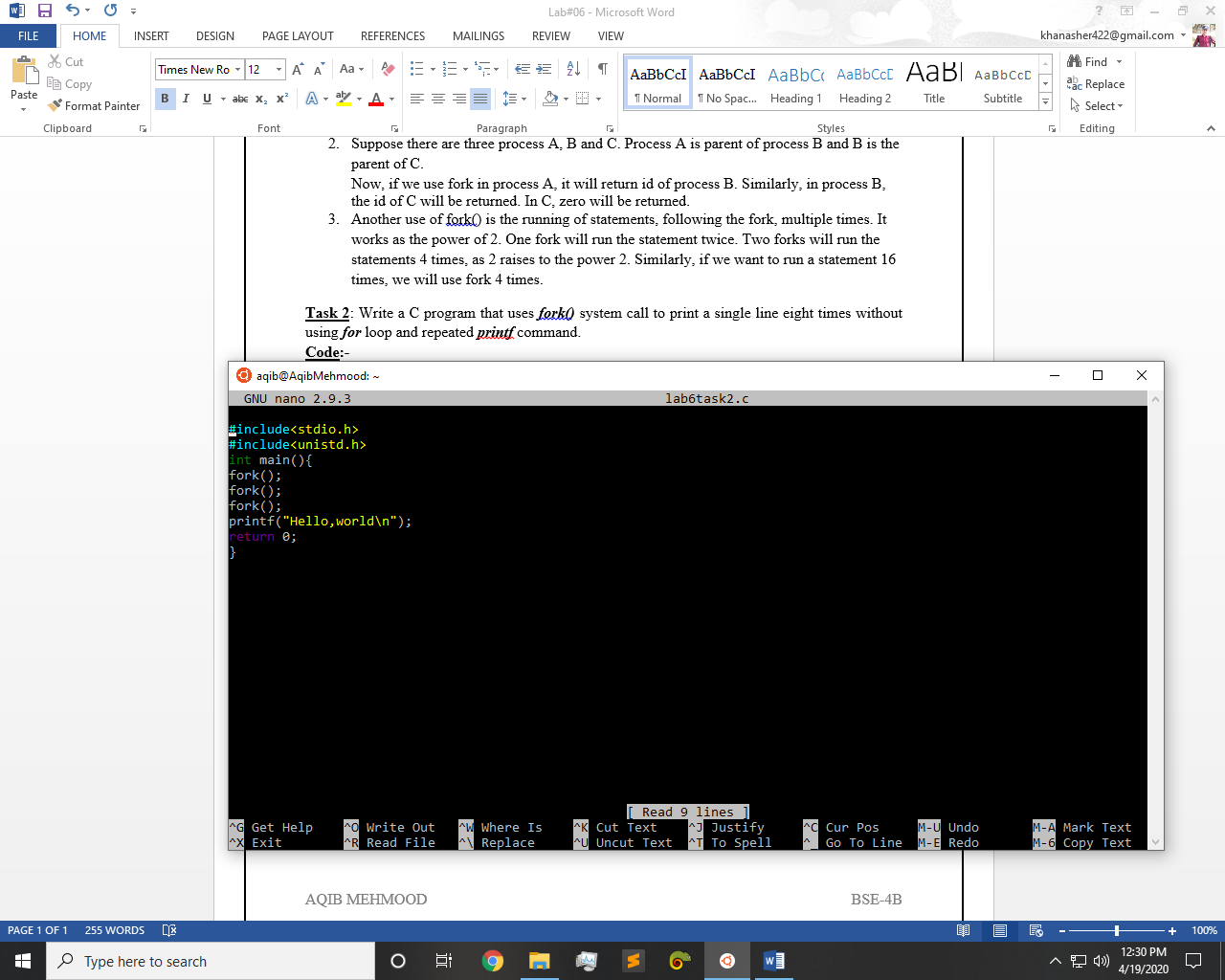
1. Fork() method is used to know about the parent and child process of current process. Fork() returns **0** if it is in a child class otherwise if it is in parent process it will return its child process’s **id**. If there is no child of the current process, it will return **-1**.
2. Suppose there are three process A, B and C. Process A is parent of process B and B is the parent of C.

Now, if we use fork in process A, it will return id of process B. Similarly, in process B, the id of C will be returned. In C, zero will be returned.

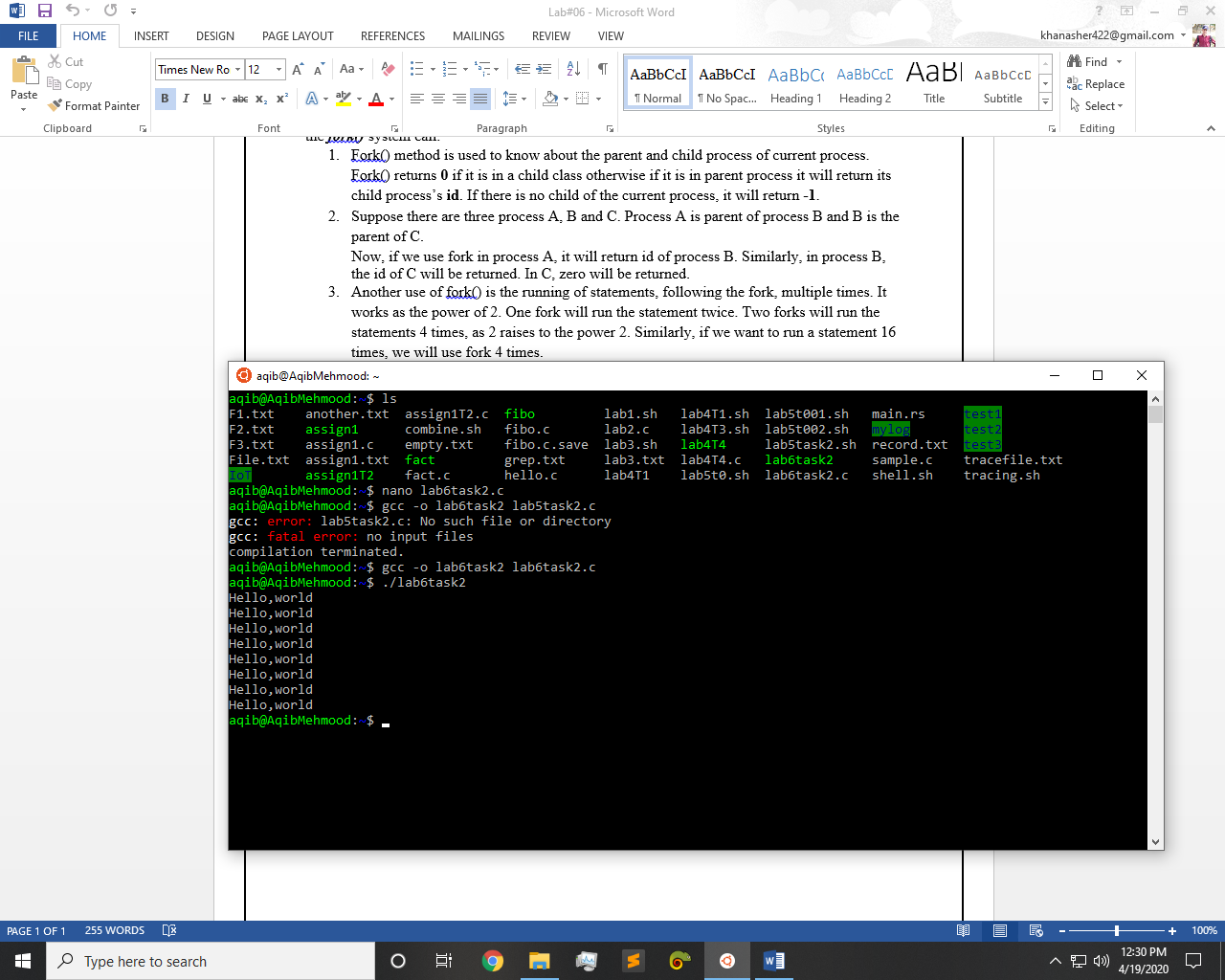
1. Another use of fork() is the running of statements, following the fork, multiple times. It works as the power of 2. One fork will run the statement twice. Two forks will run the statements 4 times, as 2 raises to the power 2. Similarly, if we want to run a statement 16 times, we will use fork 4 times.

**Task 2**: Write a C program that uses ***fork()*** system call to print a single line eight times without using ***for*** loop and repeated ***printf*** command.

**Code:-**



**Output:-**



**Task 3**: Code the C program given below and explain what it does along with providing a snapshot of the output. Investigate and write about the usage of ***execlp()*** system call.

**Execlp():-**

The execlp system call duplicates the actions of the shell in searching for an executable file if the specified file name does not contain a slash (/) character. The search path is the path specified in the environment by the path variable. If this variable isn’t specified, the default path  is used.

The execlp system call can be used when the number of arguments to the new program is known at compile time. If the number of arguments is not known at compile time, use execvp.

**Code:-**

#include <stdio.h>

#include <string.h>

// Required by for routine

#include <sys/types.h>

#include <unistd.h>

#include <stdlib.h>

int main()

{

  int pid;

 pid = fork();

  if (pid < 0)

  { // error occurred

     fprintf(stderr, "Fork failed!\n");

     exit(-1);

  }

  else if (pid == 0)

  { // child process

     printf("I am the child, return from fork=%d\n", pid);

     execlp("/bin/ls", "ls", NULL);

  }

  else

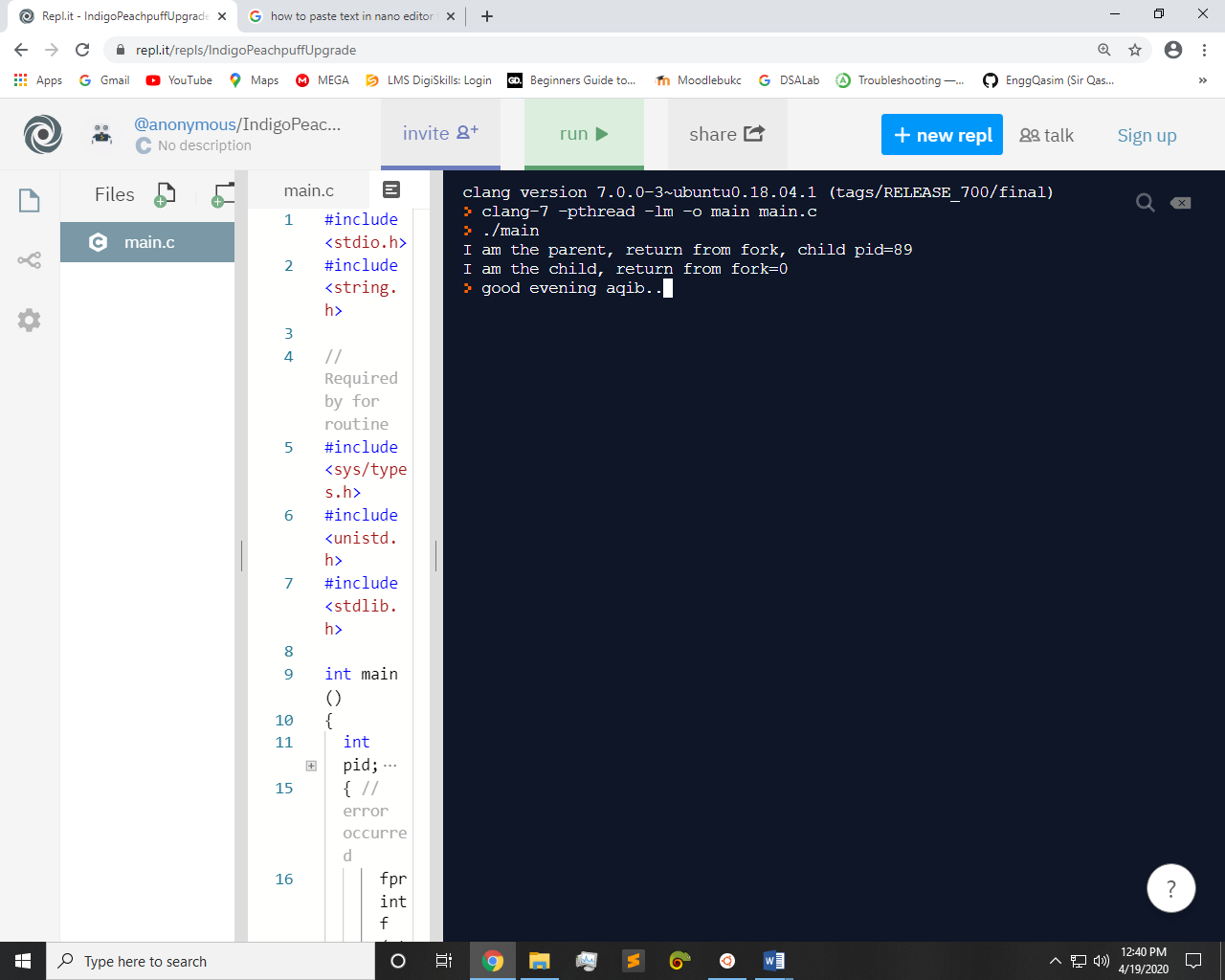
  { // parent process

     printf("I am the parent, return from fork, child pid=%d\n", pid);

  }

}

**Output:-**



~~~~~~\*\*/**THE END**/\*\*~~~~~~