Assignment 1

Prime Factorization

Introduction:

Prime factorization forms the basis of many cryptographic algorithm. Through this assignment we will understand how parallel computing can reduce the factorization time.

Objective:

To exercise students in parallel programming through the use of fork system call and use of file handling functions.

Input:

A suitably large integer , say “n”, where the number is a product of exactly two prime numbers

Output:

- The two prime factors of the input integer.
- Time taken to execute the program.

1a. Using a single process: (15 marks)

Description:

- Iterate to find the two prime factors of *n*.
- Print the result.

Example:

Input: 77

Output: Prime factors = 7, 11

Time = 1 microsecond.

1b. Using multiple processes: (25 marks)

Additional Input:

- an integer m, indicating number of processes to fork.

Description:

- Create (fork) m processes.
- Each process shall search for a prime factor in an almost equally divided range simultaneously.
- Print the result.

Example:

Input:

Number = 497503
Processes = 10

Output:

Prime Factors = 499, 997
Time = 1 microsecond.

Counting vowels and consonants

Introduction:

Data Analytics often involves processing large number of huge files. This takes a lot of time. The time can be reduced using parallel processing. This assignment illustrates how parallel programming can be used to efficiently solve problems of this genre.

Objective:

To exercise students in finding parallelism in a problem and using multiple processes to efficiently solve the problem.

Input:

One or more text (.txt) files.
sample input : - ./a.out inFile1.txt inFile2.txt inFile3.txt

Output:

- Total number of vowels in the input files.
- Total number of consonants in the input files.
- Time taken to execute the program.

1c. Using a single process: (15 marks)

Description:

- Take input i.e. file names separated by spaces as command line arguments. It is assumed that the filenames don't contain whitespaces or other special characters.
- Count the number of vowels and consonants in each file.
- Find the total number of vowels and consonants by summing up corresponding values from the individual files
- Print the total number of vowels and consonants in all the input files considered together.

1d. Using multiple processes: (25 marks)

Description:

- Take input i.e. file names separated by spaces as command line arguments. It is assumed that the filenames don't contain whitespaces or other special characters.
- Use multiple processes to count number of vowels and consonants in all the files taken together.
- Each process writes the output to a file. The filename should be the process id of the process writing the output, along with the extension 'txt'. Dont keep any other files with the extension txt.
- Now, the main process reads all the files inside the directory and calculates the total number of vowels and consonants in all the files.

Example:

Input in form of command line:

```
./a.out Trains.txt Names.txt
```

Output:

```
Total number of vowels         = 1000.
Total number of consonants = 2000
```

Comments and Indentation (20 marks)

Deliverables:

A .tar file with filename as <YOUR ROLL NO>_A1.

It shall contain two directories , ‘prime and ‘count’ for the problems 1c and 1d respectively.

It is required to have a makefile for both the problems, as the same will help to compile all the files in one go.

To run the program user shall only need to type:

```
make INPUT_FILES=File_1.txt\ File_2.txt\ File_3.txt
```