Contents

1 Running the Program

The Matrix Multiplication Program was written in Python 3.8. There is no graphical user interface implemented and everything is command-line-based. The program takes in a configuration file, does the calculations, and reports them back to the command-line interface.

1.1 Command Line Arguments

The program requires a configuration file with process details in order to run, if it is not provided, the program will run with default configurations. There are no flags required and the configuration file only changes the default functioning of the program.

The configuration file contains the following settings

MEMORY_MAX	Max amount of memory available to the system in KB
PROC_SIZE_MAX	Maximum size a process can be in KB, this is used in random number generation
NUM_PROC	The number of processes to be generated and allocated in the program
MAX_PROC_TIME	Maximum time a process can be take in ms, this is used in a random number generation for the proce

Key and descriptions of the variables in the configuration file

1.2 Configuration File Structure

The program has built in default settings to run it instantly. As mentioned previously, these default setting can be changed with a configuration file.
Here is what the default settings are set to:

```
"MEMORY_MAX" : 1024
"PROC_SIZE_MAX" : 512
"NUM_PROC" : 10
"MAX_PROC_TIME" : 10000
```

The structure of the configuration file is similar except that it uses ⟨key⟩= ⟨value⟩pairs instead of using a dictionary style.
Here is an example of how to write a configuration file:

```
<MEMORY_MAX> = <2048>
<PROC_SIZE_MAX> = <51>
<NUM_PROC> = <12>
<MAX_PROC_TIME> = <1001>
```

1.3 Main Program

To run the program, run the python file **main.py** like you would any other python file. If you want to add your own configuration then run the python file followed by the configuration file name.
The most basic example would be this:

```
python3 main.py
```

With a configuration file, it may look like this:

```
python3 main.py "config.txt"
```

2 Implementation Details

This is a simple functional python program that takes in a number of configurations and simulates the first-fit algorithm of process allocation in memory. The program uses the configurations to randomly generate a number of processes and attempt to allocate them, along the way it keeps track of memory holes created by this process, and keeps track of what portions of the memory are allocated and how long they will be allocated for. Finally, a number of statistics are collected along the way to be shown to the user. This program was only required to use one algorithm per the specifications of project 3 and as such there are no comparisons of the various approaches to process allocation.

3 Contributions

We started the project by getting together and drawing out the problem. We used a problem-solution model to approach this project. Each of the tasks was divided up among the three of us and the entire project was done in one group meeting session.

David: Programmed the entire project from scratch with the help of Himanshu and Fahad. The project was pretty easy hence why only member did the programming part.

Fahad: Attempted an implementation gathering vital information on how we should approach this problem and helped David along the way with understanding the problem.

Himanshu: Also attempted an implementation which was very helpful for reference in our final draft, helped David think through the problem.

4 Conclusion

This project was designed to variable process allocation and collect stats.

Overall, this project was a great learning experience to learn how to code in Python and how memory allocation works, especially with how memory holes can be combined. It was a great deal of fun to think through the problem s