**OPERATING SYSTEMS: THREADING PROGRAMMING ACTIVITY**

**A Documentary Research Paper Presented to**

**Computer Studies Department of College of Science**

**TECHNOLOGICAL UNIVERSITY OF THE PHILIPPINES**

**In Partial Fulfillment of the**

**Requirements for**

**OPERATING SYSTEMS**

**BSCS 2C-M**

**Manansala, Rhea Miguela M.  
Morales, Carlos Miguel T.**

**Torres, Paul Adrian O.**

**PROF. PEGARINO AMADOR, Jr.**

**JUNE** **2023**

1. **RATIONALE**

**Threading** is a programming concept that allows multiple threads of execution to run concurrently within a single process. In simpler terms, it involves creating and managing multiple independent flows of control, known as threads, within a program. Threads can be used for various purposes, such as handling user input while performing background tasks, implementing parallel algorithms, improving responsiveness in graphical user interfaces, and managing concurrent access to shared resources.

1. **Fibonacci** – Calculating Fibonacci numbers involves recursive computations, where each number depends on the two preceding numbers. By using threading, you can divide the Fibonacci sequence into smaller chunks and assign them to different threads. Each thread can independently calculate a portion of the sequence, allowing parallel execution and speeding up the overall computation. This can be particularly beneficial when dealing with larger Fibonacci numbers, as the computation time can be significantly reduced.
2. **Square** – When you need to calculate the square of a large set of numbers, threading can be utilized to distribute the workload among multiple threads. Each thread can take a subset of numbers and compute their squares concurrently. By parallelizing the calculations, you can leverage the available processing power of modern multi-core CPUs and perform the square calculations more quickly. This can be especially useful when dealing with many elements or when performing square computations repeatedly.
3. **Sort Numbers** – Depending on the sorting algorithm used, threading can be employed to divide the sorting process into multiple independent steps. For example, in a parallel merge sort, the initial array can be divided into smaller sub-arrays that are sorted concurrently by different threads. Then, the sorted sub-arrays can be merged in parallel to produce the final sorted array. This approach allows for efficient parallelization of the sorting process, which can significantly speed up the sorting of large datasets.
4. **ALGORITHM, CODE AND RUN PROGRAM**

**CODE FOR MAIN**

A picture containing text, screenshot, font

Description automatically generated

**CODE FOR FIBONACCI**

**A screen shot of a computer program

Description automatically generated with medium confidence**

**CODE FOR SQUARE**

**A picture containing text, screenshot, software, multimedia software

Description automatically generated**

**CODE FOR SORT NUMBERS**

**A screen shot of a computer program

Description automatically generated with low confidence**

**RUN PROGRAM**

**A screenshot of a computer program

Description automatically generated with medium confidence**