# Asst 02: Spooky Searching Testplan

## 1 Introduction

### 1.1 Definitions:

A **testsuite** is a group of executions with varying inputs.
An **execution** is defined as a search done - for a random value - a given amount of times and returns a batch result.
A **batch result** is a struct containing five variables of type long. These values are the total time for the execution to run, the average time for one search, the minimum time for one search, the maximum time for one search, and the standard deviation for all searches within the execution.
The **SIZE**, also denoted as **n**, is the size of the array containing the random values.
The **SPLIT SIZE**, also denoted as **ns**, is the size of each split that each process or thread will run through.

### 1.2 Procedure:

We will write functions that will perform multiple executions while varying the **SPLIT SIZE** and the **SIZE**. We then observe critical points of interest, and will discuss them later. We also have a basic sequential search execution method, which we will compare with the thread and process search methods.
*These functions will be run multiple times on different days to ensure a more accurate average time value.*

## 2 Testsuites

### 2.1 Testsuite 1: *Determining the integrity of our code.*

**Objective:** Ensure that our code is actually working as expected (*i.e. A search for a value returns the expected index*).
**Method:** We begin this testsuite with an **n** of 1000 and **ns** of varying size (*The ns in this testsuite is a predefined macro. We have run this testsuite with multiple different values of the macro.*)

### 2.2 Testsuite 2: *Varying n (SIZE) while keeping ns (SPLIT SIZE) constant.*

**Objective:** Determine the relationship between efficiency (time) and **n**.
**Method:** We begin this testsuite by running an execution with variables **n** of 10, and an initial **ns** of the value predefined by the macro. We then begin to increase the size of **n** by 10 until we reach an **n** of 1000. Throughout this procedure, **ns** is constant.
We will run the testsuite and observe the results for types proc, thread, and sequential.

### 2.3 Testsuite 3: *Keeping n (SIZE) constant while varying ns (SPLIT SIZE).*

**Objective:** Determine the relationship between efficiency (time) and **ns**.
**Method:** We begin this testsuite by running an execution with variables **n** and and initial **ns** both predefined by macros. We then run multiple executions while increasing the **ns** by 10 until a value of 250 is reached. The **n** is kept constant throughout this testsuite at 1000.
We will run the testsuite and observe the results for types proc, thread and sequential.

### 2.4 Notes regarding the testsuites:

- Both our **n** and **ns** initial values are macro defined. We have run all testsuites multiple times at multiple different macro defined values of **ns**.
  The **ns** value in any given run of searchtest is to be one of the following: {10, 20, 30, 40, 50, 75, 100, 125, 150, 175, 200, 225, 250}.
- Similarily, but only for testsuite 3, the **n** value at any given run of searchtest is to be one of the following: {100, 1000, 2000, 5000, 10000, 20000}.
- There exists a rescrambling process between each testsuite run. This is to emulate a "new" random list of numbers.

## 3 Conclusion

We believe these testsuites described will allow us to develop a strong thesis on the relationship between the **SIZE**, **SPLIT SIZE**, and time for traversing an array with processes, with threads, and with neither.