



1. Create two integer arrays A and B. The two arrays must contain at least 256 elements but must be of unequal size. Populate both arrays with randomly generated integers between 0 and 1024. Sort array A using Shell Sort and array B using Quick Sort.
2. Take the two sorted arrays A and B from the question above and merge into a new array C. You must do this in linear time. Note that the size of C must be size of A plus size of B.
3. Let A be an array of n elements (that is, the elements of A are  $A[0], \dots, A[n-1]$ ). An element  $A[i]$  is **extreme** if the following conditions hold regarding  $A[i]$ .
  - $A[i]$  is not the first nor the last element of A. That is,  $0 < i < n - 1$  and either  $A[i - 1] < A[i] > A[i + 1]$  or  $A[i - 1] > A[i] < A[i + 1]$ . For example, the extreme points of an array [0, 5, 3, 6, 8, 7, 15, 9] are 5, 3, 8, 7, 15.

Write an algorithm that prints the extreme points of the given array. If there are no extreme points, the algorithm prints "SORTED". Do you agree that an array has no extreme points if and only if it is sorted? Explain your answer.

4. Write a program that, given a list of integers, finds all 2-pairs of integers that have the same product. A 2-pair is 2 distinct pairs of integers  $((a,b),(c,d))$  where  $a \times b = c \times d$  and  $a \neq b \neq c \neq d$ . The range of integers in the list should be from 1 to 1024.
5. Write a program that uses an ADT Stack to evaluate arithmetic expressions in RPN format. The contents of the stack should be displayed on the screen during evaluation. The allowed arithmetic operators are +, -, x, and /.
6. Write a Boolean function that checks if a number is **prime**. Also implement the Sieve of Eratosthenes algorithm. Explain any optimizations made.
7. Write a program that accepts as input a sequence of integers (one by one) and then incrementally builds, and displays, a Binary Search Tree (BST). There is no need to balance the BST.

8. Write a program that finds an approximation to the **square root** of a given number *n* using an iterative numerical method such as the *Newton-Raphson Method*.
9. Write a program that, given an array of integers, finds all integers in the array that are repeated more than once. Try to find the fastest and most memory-efficient way of doing this.
10. Write a **recursive** function that finds the largest number in a given list of integers.
11. Write a function that computes **cosine** or **sine** by taking the first *n* terms of the appropriate series expansion.
12. Write a function that returns the sum of the first *n* numbers of the Fibonacci sequence (Wikipedia). The first 2 numbers in the sequence are 1,1,...

**Deadline is *Wednesday 29<sup>th</sup> May, 2019 at Noon*. Do not forget the Statement of Completion and the Plagiarism Declaration Form. Submit the printed coursework to the secretary of the Department of CIS. CDs and DVDs for documentation, code and binaries – no pendrives please. You must also upload the assignment, as a single Word or PDF file to Turnitin.**

**Allowed programming languages are Python, C, C++, Objective C, Swift, Go, Pascal, Java, Javascript, Haskell, F#, and C#. If you want to use another programming language please email me. *You cannot mix languages!***

**Your printed (and PDF/Word) coursework document must include the source code, sample screen dumps, and explanation of how each program was tested.**

**The Statement of Completion is a list of which questions were attempted, which work, and which have bugs. For example:**

- **Question 1 – Attempted and works well.**
- **Question 2 – Not attempted.**
- **Question 3 – Attempted but does not work.**
- **Question 4 – Attempted but has the following bug...**
- **Etc.**

**The Statement of Completion, like the Plagiarism Declaration, must be signed.**