University of Nottingham
UK | CHINA | MALAYSIA

Lab Worksheet 5 (17/03/2025) CELEN087

Instruction:

Tutor will demonstrate the complete process of designing MATLAB functions based on the classical methods (algorithms) used in mathematics and computer science.

Programming Examples:

1. Design a MATLAB program that implements the Binary Search Method.
   (a) Quick review: what is the principle of this method?
   (b) Analyze the programming task and attempt the code
      - Design input/output variables: how many? what types?
      - Design control flows: which part can be repeatedly processed? by using what structures (for/while loops, recursion)? when to stop?
      - Design supporting variables: what other variables need to be created along the process? what are the initial values of them?
      - Design statements: In the repeated process, what variables need to get the values updated? how to update them?
   (c) Test and debug the code
      - Check the logic in your code: does it exactly follow the idea of the method?
      - Execute the code: any syntax error reported? how about after using another example?
      - Check the correctness: does it always return the correct result? any extreme cases we need to consider?
   (d) Finalize the program
      - Package the program into a function.
      - Test the function.
      - Add necessary comments.
      - Any other demand? How to modify the function accordingly?
2. Design a MATLAB program for finding the root of $f(x) = x^2 - \sin x$ on the interval $[0.5, 1]$ using Bisection Method with an error tolerance 10^{-5}.
3. Design a MATLAB program that sorts a list of numbers into ascending order using Bubble Sort Method.

Related Exercises:

4. (a) Write a MATLAB function **linearSearch** that can search for a key value from an array of any length. It should return 1 if the key is found in the array, otherwise return 0. For example,

   `linearSearch(1,[5, 20, 14, 3])` = 0
   `linearSearch(3,[5, 20, 14, 3])` = 1
   `linearSearch(1,[ ])` = 0

   (b) Modify your function **linearSearch**, so that if the key is found, your function should return the index number of the key in the array (let's assume that there are no repeated numbers in the array); if the key is not found, your function should return -1. For example,

   `linearSearch(3,[5, 20, 14, 3])` = 4
   `linearSearch(5,[5, 20, 14, 3])` = 1
   `linearSearch(2,[5, 20, 14, 3])` = -1
   `linearSearch(1,[ ])` = -1

   (c) Now let's assume the array may contain repeated numbers in it. Write a MATLAB function **countRec** that finds the reccurence (frequency) of a key number appearing in the array. For example,

   ```
   countRec(5,[2, 9, 1, 4, 1, 5, 5, 1, 1]) = 2
   countRec(1,[2, 9, 1, 4, 1, 5, 5, 1, 1]) = 4
   countRec(6,[2, 9, 1, 4, 1, 5, 5, 1, 1]) = 0
   countRec(5,[ ]) = 0
   ```

5. Write a MATLAB function **myBubbleSort** that follows the idea of Bubble Sort Method, but rearranges the array into descending order. For example,

   `myBubbleSort([2, 5, 1, 9, 8, 2, 9]) = [9, 9, 8, 5, 2, 2, 1]`

6. Write a MATLAB script **myTest** that tests your functions by doing the following:

   - Create an integer variable `key`, that is randomly generated between 1 and 5.
   - Create an array `randArray`, with 20 randomly generated integers between 1 and 20.
   - Check if the `key` exists in the `randArray` using **linearSearch**.
   - Find the reccurence of the `key` using **countRec**.
   - Sort the array into `sortedArray` using **myBubbleSort**.
   - Check if the `key` exists in the `sortedArray` using **binarySearch**.