Addis Ababa institute of Technology

Center of Information Technology and Scientific Computing

Fundamentals of computer science and programming

Lab 09: Data Structures

Exercises

1. (Linear Search)
   In Linear Search, we sequentially iterate over the given list and check if the element we are looking for is equal to the one in the list. Some quick points about Linear Search.
   Define a function that finds a given input from a list. The function should return the index of the found input. If the input does not reside in the list return -1

   Tests:

```
find ([1,2,4], 4)                    # should return 2
find ([ 'hi' , ' there' ],' hi' )     # should return 0
find ([ 'one' , 2,  'there' ], 1)     # should return -1
```

2. Write a function that finds the first appearance of given keyword in a sentence given the sentence as one parameter and the keyword as another.
   (Clue use <str>.split() reuse the function written for the first question)

   Tests:

```
find_word('paramount sees green in the red, 'paramount')   # should return 0
find_word('paramount sees green in the red, 'para')        # should return -1
```

3. (Binary Search) Binary Search is a search algorithm that finds the position/index of an element within a sorted search list.

   Let's say that we have to search the index of `32` in `[4, 8, 9, 10, 24, 32, 45, 56]`

   - Look for the middle element in the list. It is `24`.
   - Compare that middle element with `32`. `24` is less than `32`.
   - This means that all the numbers on the left of `24` are less than `24` and there's no point of searching for `32` in that part of the list.
   - So we will look in the right half of `24`.
   - Now, our list looks like this. [~~4, 8, 9, 10, 24,~~ 32, 45, 56].
   - Again, look for the middle element, `45`. `45 > 32`. So no point in looking at elements in the right half of `45`.
   - Now the list has broken down to [~~4, 8, 9, 10, 24,~~ 32, ~~45, 56~~], the only element left is `32`. we have found what we were looking for.

4. (Recursion) Write a recursive function that checks if a word is palindrom

5. (Back to the DNA sequence complement) Modify the program that was written in the previous lab to determine the second chain of a DNA sequence. Choose a good data structure such that there are NO if/else construct.

6. Build a function that records the number of words in a given list and returns the counter dictionary

   >>> count_words(['the', 'clown', 'ran', 'after', 'the', 'car', 'and', 'the', 'car', 'ran', 'into', 'the', 'tent', 'and', 'the', 'tent', 'fell', 'down', 'on', 'the', 'clown', 'and', 'the', 'car'])

   Return Value: {'and': 3, 'on': 1, 'ran': 2, 'car': 3, 'into': 1, 'after': 1, 'clown': 2, 'down': 1, 'fell': 1, 'the': 7, 'tent': 2}

7. Modify the function you have written in q06 such that it accepts a sentence as a parameter and determines the frequency of each word in the sentence.