Checkpoint 2 review

CS51 – Spring 2026

Basic info

Checkpoint 2

- In class, on Tuesday April 14th
- Can bring a double-sided handwritten (ok if with a stylus) sheet of paper with notes
- Will cover everything in Mathematical Foundations and Data Structures and Algorithms units

Topics

- Propositional Logic (logical connectives and their truth tables), tautology, satisfiable
- Loop invariants (pre-, post-conditions, all four steps to show full correctness of iterative programs)
- Proof by induction (both weak and strong)
- Recursion (comfortable with writing a recursive program and reading recursive code in Python)
- Turtle module (basic commands to control)
- Sequences (lists, strings, tuples, ranges) and dictionaries. Comfortable with indexing, slicing, membership, iterating
- Dictionaries
- Nested lists, list comprehension, files
- Sorting algorithms (bubble, selection, insertion sort) basic idea and complexities

Practice Problem 1

- Consider the following pseudocode, which operates on a string counts the number of vowels:

```
i = 0
count = 0
while i < len(my_string):
    if my_string[i] is a vowel:
        count +=1
    i += 1
```

- Prove that it's correct using loop invariants

Answer 1

- Basic idea is to show loop invariant: at the beginning of the i-th iteration, `count` is equal to the number of vowels in `my_string[0:i]`
- In maintenance, you want to consider two cases:
  - Case 1: `my_string[i]` is a vowel, then `count` correctly increases and given assumption now reflects the number of vowels in `my_string[0:i+1]`
  - Case 2: `my_string[i]` is *not* a vowel, then `count` does not change and given the assumption now reflects the number of vowels in `my_string[0:i+1]`

Practice Problem 2

- Using weak induction, prove that for all non-negative integers n, $3 | (n^3 - n)$ that is that $n^3 - n$ is evenly divisible by 3.

Answer 2

- Let $P(n)$ be the claim that $n^3 - n$ is divisible by 3, that is there is a constant m such that $n^3 - n = 3m$.
- **Base case**: For $n = 0$, we have $P(0) = n^3 - n = 0^3 - 0 = 0 = 3 \times 0$. Hence $3|0^3 - 0$ and the base case holds.
- **Inductive case**: Let n be an arbitrary non-negative number and assume $3 \mid (n^3 - n)$. By the definition of divisibility, there is a non-negative integer m such that $n^3 - n = 3m$.
- We will prove that $P(n+1)$ holds. Indeed
- $(n+1)^3 - (n+1) = (n^3 + 3n^2 + 3n + 1) - n - 1 = (n^3 - n) + 3n^2 + 3n = 3m + 3n^2 + 3n = 3(m + n^2 + n) = 3m'$, that is there is a constant m' such that $(n+1)^3 - (n+1) = 3m'$
- Thus, we have proven $P(n+1)$ and by the principle of mathematical induction, we have proven $P(n)$ for all non-negative integers.

Practice Problem 3

- What does the following function do (in one sentence) assuming x is a list:

```
def mystery(x):
    if len(x) <= 1:
        return True
    else:
        if x[0] > x[1]:
            return False
        else:
            return mystery(x[1:])
```

Answer 3

- Checks if the list is sorted in ascending order.

Practice Problem 4

- Write a function called swap that takes as input a dictionary and returns a new dictionary where key is the old value and the value is the old key, i.e swaps the key/values. You may assume that the value are unique.

- >>> d = {"a":1, "b":2, "c":3}
- >>> swap(d)
- {1: 'a', 2: 'b', 3: 'c'}

Answer 4

- Write a function called swap that takes as input a dictionary and returns a new dictionary where key is the old value and the value is the old key, i.e swaps the key/values. You may assume that the value are unique.

```
def swap(d):
    new_d = {}
    for key in d:
        value = d[key]
        new_d[value] = key
    return new_d
```

Practice Problem 5

- Write a function called `capitalized_lines` that takes as input a file and counts the number of lines that start with a capital letter.

Answer 5

- Write a function called capitalized_lines that takes as input a file and counts the number of lines that start with a capital letter.

```
def capitalized_lies(filename):
    file = open(filename, 'r')
    count = 0
    for line in file:
        if line[0].isupper():
            count += 1
    return count
```