MATHEMATICAL NOTATION

Square brackets are inclusive, curve brackets are not inclusive

Do not get this mixed up!

[0…5] = 0, 1, 2, 3, 4, 5

(0…5) = 1, 2, 3, 4

[0…5) = 0, 1, 2, 3, 4

(0…5] = 1, 2, 3, 4, 5

[0…-1] = Empty list

[0…0] = Only 1 element in the list

INVARIANT

Loop invariant

Something that doesn't change in your loop that will lead to the outcome

Initialization

Proving that the invariant holds at the beginning before entry of loop

Maintenance

Proving that the invariant holds true throughout the algorithm

Termination

Proving that the invariant implies correctness of algorithm after termination upon exit of loop

2021 Semester 02

In the following 4 questions, you will write a correctness proof of the algorithm **max_five** given below. You must use the **invariant-initialization-maintenance-termination** structure taught in this unit. If you do not follow this structure, you may receive no marks for your proof.

The following code **calculates the difference between the sum of all the even numbers and the sum of all the odd numbers** in L (i.e. it subtracts the sum of the odd numbers from the sum of the even numbers)

```
def f(L):
  i = 0
  s = 0
  while i < len(L):
    if L[i] % 2 == 1:
      s -= L[i]
    else:
      s += L[i]
    i += 1
  return s
```

#loop invariant

Question 1

Write the useful invariant for the function. Remember that to be a useful invariant, we have to be able to complete a correctness proof using this invariant.

Be careful of off-by-one errors!

2 Marks

Question 2

Show that the invariant you gave in question 1 holds **before the first time that the loop condition is checked.**

1 Mark

Question 3

Show that the invariant holds just **before the loop condition is checked in each iteration** of the loop. Your proof must **follow the structure given** in the unit.

2 Marks

Question 4

Argue that the algorithm is **correct** using the **termination condition** of the algorithm.

1 Mark

1) s is the difference between the sum of all the even numbers and the sum of all the odd numbers in L[0…i-1]. The reason it is i-1 and not i is because we have not processed the i^{th} element yet as the invariant is at the start of the loop not at the end.

2) Before the first time that the loop condition is checked, we can see that our i = 0, s = 0. Because of this and the invariant we gave in question 1, we know that before the first loop condition, s is the difference between the sum of all the even numbers and the sum of all the odd numbers in L[0…0-1], which is L[0…-1]. The difference between the sum of all even numbers and sum of all odd numbers in an empty list is 0, and before the first loop condition is checked, s is indeed 0, hence why the invariant in question 1 holds.

3) In the loop, there are 2 cases. First case is when the i^{th} element is odd, and the second case is when the i^{th} element is even. In the case that i^{th} element is odd, $s_{new} = s – L[i]$. In the case that i^{th} element is even, $s_{new} = s + L[i]$. Therefore, at the end of the loop, s is the difference between the sum of all the even numbers and the sum of all the odd numbers in L[0…i].

4) The loop terminates when i = len(L). Upon exit of the loop, s is the difference between sum of all the even numbers and the sum of all the odd numbers in L[0…(len(L)-1], which is equivalent to L[0…N-1], where N is len(L). This proves that the invariant holds true for the entire list.

2021 Semester 01

Question 1

Consider the following algorithm to **determine the parity of the sum of a list**.

Note that *not 0 = 1, not 1 = 0*.

1 Mark

```
def parity(L[1..n])
  p = 0
  i = 1
  ###here###
  while i <= n
    if L[i] % 2 == 1
       p = not p
    i += 1
  return p
```

#loop invariant

Write down a **useful** invariant for the algorithm above. Your invariant **must** be true at the line marked with the comment "###here###", i.e. just before the check of the while condition.

Question 2

Show that the **invariant you identified in Question 1 holds** at the line marked #Here, **before the loop is entered for the first time**.

0.5 Mark

Question 3

Show that your **invariant is true** at the line marked ###here### each time the loop runs (excluding *i=1*, since you will have **proved that in Question 2**.

3 Marks

Make sure your argument is logically sound.

Question 4

Since you have now shown that the invariant you chose in **Question 1** holds for all iterations of the loop, now argue that **the algorithm is correct**.

0.5 Mark

1) p is the parity of the sum of a list L[1…i-1]. The reason it is i-1 and not i is because we have not processed the i^{th} element yet.

2) Before the first time that the loop condition is checked, we can see that our p = 0, i = 1. Because of this and the invariant we gave in question 1, we know that before the first loop condition, p is the parity of the sum of a list L[1…1-1], which is equivalent to L[1…0]. This is true because the parity of the sum of an empty list L[1…0] is 0. Before the first loop condition, p is 0, which is indeed true hence why the invariant in question 1 holds.

3) At the end of each loop, the parity of the sum of the list only flips if the i^{th} element is an odd number. Since we know that from question 1 the loop invariant is that p is the parity of the sum of a list L[1…i-1], then to make it hold for L[1…i] we just have to flip the parity if the i^{th} element is odd, else the parity stays as it is. Therefore, at the end of each loop, p is the parity of the sum of a list L[1…i].

4) Since the invariant holds at every loop as shown in question 3, it must hold for the last loop as well. From the algorithm, we see that the loop terminates when i = n+1. Upon exit of the loop, p is the parity of the sum of a list L[1…(n+1)-1], which is equivalent to L[1…N]. This proves that the invariant holds true for the entire list.