Задача 1:

Ако searched **е** елемент на масива, то **linearSearch** връща индекса на първото му срещане.

Ако searched **НЕ Е** елемент на масива, то **linearSearch** връща -1.

Инварианта: За всяка проверка за край на цикъла е изпълнено:

в подмасива **arr[0...i-1]** не се съдържа елементът searched

База: При първата проверка: i = 0.

arr[0 ... i-1] = **arr[0 ... -1]** - празният подмасив. searched **не е** елемент на празния подмасив. **ОК!**

Поддръжка: Допускаме, че инвариантата е изпълнена за някоя проверка за край на цикъла

,която не е последна.

Т.е. В arr[0...i-1] не се съдържа searched.

Щом проверката за край на цикъла не е последна, то условието в if-а на ред 9 ще даде лъжа.

-> arr[i] не е searched.

1) searched го няма в arr[0...i-1].

2) arr[i] **не** е searched.

1) & 2) -> searched не е елемент на подмасива arr[0...i]

Но веднага след това i се инкрементира.

-> searched не е елемент на arr[0...i-1]. **ОК!**

Терминация: Цикълът може да приоключи от 2 места:

1сл. цикълът приключва на ред 10 (return i). Тогава от инвариантата **searched не е елемент** на arr[0..i-1].

Но щом цикълът приключва на ред 10, то проверката на ред 9 е върнала истина. Т.е searched = arr[i].

-> **i** е индексът **на първото срещане** на searched в arr. На ред 10 връщаме точно **i**. **ОК!**

2сл. цикълът приключва при i = len. Тогава от инвариантата: searched не е елемент на arr[0..i-1] = arr[0..len-1].

Но това е целият масив! Т.е searched не елемент на масива arr. На ред 12 връщаме -1. **ОК!**

Задача 2:

Алгоритъмът връща **|_sqrt(n)_| (закръглено надолу)**

1сл. Ако n = 0 или n=1, то sqrt(n) = n. На ред 7 връщаме точно n.

2сл. Ако n > 1.

Инварианта: За всяка проверка за край на цикъла е изпълнено: **res = i^2 & (i-1)^2 <= n**

База: При първата проверка за край: res = 1, i = 1

res = 1^2 ОК! (1 - 1)^2 <= n **OK!**

Поддръжка: Допускаме, че инвариантата е изпълнена за някоя проверка за край, която не е последна.

Т.е: res = i^2 (i-1)^2 <= n.

Щом проверката не е последна: res <= n -> i^2 <= n.

Нека с **i'** и **res'** бележим новите стойности на променливите **i** и **res**.

1) **i'** = **i** + 1

2) **res'** = **i'** * **i'** = (**i** + 1)^2.

Вярно ли е, че: **res'** = i'*i' и (**i'** -1)^2 <= n.

1) очевидно. **OK!**

2) (**i'** -1)^2 = (**i** + 1 - 1)^2 = **i**^2 <= n (от това, че проверка за край не беше последна). **OK!**

Терминация: Цикълът ще приключи: res > n.

res > n (цикълът приключва)

res = i^2

(i - 1)^2 <= n

Следователно:

(i - 1)^2 <= n < res = i^2

i-1 <= sqrt(n) < i

-> i-1 е sqrt(n) **(закръглено надолу)**.

На ред 16 връщаме i-1. **OK!**

Задача 3:

Алгоритъмът връща **дали n е просто**.

1сл) n <= 1 -> n **не е просто**, на ред 8 връщаме **лъжа**.

2сл) n > 2

Инварианта: За всяка проверка за край на цикъла е изпълнено:

n няма делители измежду [2 ... i-1].

База: При първата проверка: i = 2. n няма делители измежду [2 ... 2-1] = [2 ..1] = []. OK!

Поддръжка: Допускаме, че инвариантата е изпълнена за някоя проверка, която не е последна.

Т.е. n няма делители в [2 ... i-1].

Щом проверката не е последна, то условието в if-а на ред 13 дава лъжа. Т.е**. i НЕ е делител на n**.

-> n няма делители в [2 ... i]. След това i се инкрементира.

n **няма делители** в [2 .. i-1] **OK!**

Терминация: Цикълът може да приключи от 2 места:

1сл) От ред 14. Следователно условието на ред 13 е било истина. -> **i е делител на n**.

i >= 2. T.e i е **нетривиален делител**. Т.е n не е просто. На ред 14 връщаме **лъжа**.

2сл) Цикълът приключва при i, където i > sqrt(n).

От инвариантата знаем, че n няма делители [2 ... i-1]. i-1 <= sqrt(n)

От **твърдение *** -> n е просто. На ред 16 връщаме истина! **OK!**

С _sqrt(n)_ бележим sqrt(n) закръглено надолу!

Твърдение *: Ако n няма делители измежду [2…_sqrt(n)_], то n е просто:

Доказателство: Допускаме противното. Т.е:
n няма делители измежду [2…n] и n не е просто.
Щом n не е просто, то n **има цял делител във вида** _sqrt(n)_ +k (k > 0, k е есествено число).
Но n / sqrt(n) + k **също е делител**. Нека e _sqrt(n)_ + t (k >0, t е естествено число).
Но (sqrt(n) + t) * (sqrt(n)+k) > n (очевидно). **Противоречие**!

- ➔ Ако n няма делители измежду [2…_sqrt(n)_], то n е просто.

Задача 4:

SampleBubbleSort е валиден сортиращ алгоритъм.

arr' съдържа същите елементи като arr, но в сортиран вид!

Твърдение за вътршния цикъл:

След изпълнение на итерациите на вътрешния цикъл:

arr[len - 1 - i] е най-големият елемент в arr[0 .. len - 1 - i].

Инварианта за вътрешния цикъл:

За всяка проверка за край на цикъла: arr[j] е най-големият елемент в подмасива arr[0..j].

База: При първата проверка: j = 0. arr[j] е най-големият елемент в подмасива arr[0..0] **ОК**!

Поддръжка: Допускаме, че инвариантата е изпълнена за някоя проверка за край, която не е последна.

Т.е. : **arr[j] е най-големият елемент в arr[0..j].**

1 сл) Проверката на ред 21 връща **истина**! arr[j+1] < arr[j]. -> arr[j] **е най-големият елемент в** arr[0...j+1].

На ред 22 **разменяме елементит**е -> arr[j+1] е най-големият елемент в arr[0...j+1].

Веднага след това j се инкрементира -> arr[j] е най-големият елемент в arr[0..j] **ОК!**

2 сл) Проверката на ред 22 връща **лъжа**!

arr[j+1] >= arr[j]. Следователно arr[j+1] е **най-големият елемент** в подмасива arr[0..j+1]

Веднага след това j се инкрементира -> arr[j] е **най-големият елемент в** arr[0..j] **ОК!.**

Терминация: Цикълът приключва при j = len - 1 - i.

От **инвариантата** знаем, че **arr[j] е най-големият елемент в arr[0...j].**

arr[len - 1 - i] е най-големият елемент в arr[0 .. len - 1 - i].!

След изпълнение на итерациите на външния цикъл:

arr съдържа същите елементи, но в сортиран вид.

Инварианта (за външния цикъл):

За всяка проверка за край на цикъла:

arr[len - i ... len - 1] съдържа най-големите елементи от arr но в сортиран вид!

База: При първата проверка за край: i = 0. arr[len ... len - 1] (**празен**) е сортиран! **ОК!**

Поддръжка: Допускаме, че инвариантата е изпълнена за някоя проверка за край, която не е последна!

arr[len - i ... len - 1] съдържа **най-големите елементи** от arr но в **сортиран вид!**

След пълно изпълнението на всички итерации на вътрешния цикъл:

1) arr[len - i - 1] е най-големият елемент в подмасива arr[0 .. len - i - 1].

От инвариантата знаем, че:

2) arr[len - i - 1] **е по-малък от елементите** в arr[len - i ... len-1] (там са най-големите елементи).

3) arr[len - i .. len - 1] е **сортиран**.

-> От 1) 2) и 3): arr[len – i - 1 ... len - 1] съдържа **най-големите елементи** от arr но в **сортиран вид!**

На следващия ред i се инкрементира.

arr[len - i ... len - 1] съдържа най-големите елементи от arr но в сортиран вид!

Терминация: При последната проверка: i = len

От инвариантата знаем, че arr[len - i ... len - 1] е **сортиран**.

arr[len - len ... len -1] е **сортиран**. **arr[0 ... len-1] е сортиран**.

Това е целият масив!

Това е сортиране чрез размени (**swap-ове**). Т.е масивът съдържа същите елементи!