Упражнение 3 (коректност на алгоритми чрез инварианта)

Пример 1: Даденият алгоритъм връща 2^n:

```
int alg1(int n)
{
    int s = 1;
l1: for (int i = 0; i < n; i++) s *= 2;
    return s;
}
```

Инварианта на цикъла: При всяко k-то достигане на `l1`: $i = k - 1$, $s = 2^{k-1}$.

База: При първото достигане: $i = 0$, $s = 2^0 = 1$ - вярно

Поддръжка: Нека е вярно за някое k-то достигане, което не е последното:
$i = k - 1$, $s = 2^{k-1}$

В тялото на цикъла имаме `s *= 2`, така че на $k + 1$-вото достигане ще имаме: $i = k$, $s = 2^{k-1}.2 = 2^k$ - твърдението отново е вярно.

Терминация: При $n + 1$-вото достигане имаме: $i = n$, $s = 2^n$.
В този момент цикълът приключва.

След цикъла имаме `return s`, което връща стойността на s или 2^n.

Пример 2: Даденият алгоритъм връща сумата на елементите на масива $a[n]$:

```
int a[n];
int alg2()
{
    int s = 0;
l1: for (int i = 0; i < n; i++)s + = a[i];
    return s;
}
```

Инварианта на цикъла:
При всяко k-то достигане на `l1`: $i = k - 1$, s съдържа $a[0] + \ ... + a[k - 2]$.

База: При първото достигане: $i = 0$, s съдържа $a[0] + \ ... + a[-1]$ - вярно

Поддръжка: Нека е вярно за някое k-то достигане, което не е последното:
$i = k - 1$, s съдържа $a[0] + \ ... + a[k - 2]$.

В тялото на цикъла имаме `s += a[i]`, което е същото като `s += a[k-1]`, така че на $k + 1$-вото достигане ще имаме: $i = k - 1$, s съдържа $a[0] + \ ... + a[k - 2] + a[k - 1]$ - твърдението отново е вярно.

Терминация: При $n + 1$-вото достигане имаме: $i = n$, s съдържа $a[0] + \ ... + a[n - 1]$.
В този момент цикълът приключва, а след него имаме `return s`, което връща сумата на елементите на $a[n]$.

Пример 3: Коректност на Selection Sort

```
int a[n];
void sel_sort()
{
      for (int i = 0; i < n - 1; i++) for (int j = i + 1; j < n; j++)
            if (a[i] > a[j]) swap(a[i], a[j]);
}
```

Инварианта на вътрешния цикъл:

При всяка k-та итерация на вътрешния цикъл $a[i]$ съдържа $\min\{a[i], \dots, a[j-1]\}$.

<u>База:</u> При първата итерация имаме:

$j = i + 1$,

$a[i]$ съдържа $\min\{a[i], \dots, a[j-1]\} = \min\{a[i], \dots, a[i]\} = a[i]$ - вярно

<u>Поддръжка:</u> Нека е вярно за k-та итерация, различна от последната:

$j = i + k \neq n$,

$a[i]$ съдържа $\min\{a[i], \dots, a[j-1]\}$.

След изпълнение на `if (a[i] > a[j]) swap(a[i], a[j])`, $a[i]$ съдържа $\min\{a[i], a[j]\} = \min\{\min\{a[i], \dots, a[j-1]\}, a[j]\} = \min\{a[i], \dots, a[j]\}$.

При следваща итерация на цикъла имаме:

$j = i + k + 1$,

$a[i]$ съдържа $\min\{a[i], \dots, a[(j+1)-1]\}$ - твърдението отново е вярно.

<u>Терминация:</u> При последната итерация имаме:

$j = n$,

$a[i]$ съдържа $\min\{a[i], \dots, a[j-1]\} = \min\{a[i], \dots, a[n-1]\}$.

Инварианта на външния цикъл:

При всяка k-та итерация на външния цикъл, елементите $a[0], \dots, a[i-1]$ са сортирани и в тях се съдържат i-тите най-малки елементи на масива.

<u>База:</u> При първата итерация имаме:

$i = 0$,

празният масив е сортиран - вярно

<u>Поддръжка:</u> Нека е вярно за k-та итерация, различна от последната:

$i = k - 1 \neq n - 1$,

елементите $a[0], \dots, a[i-1]$ са сортирани и в тях се съдържат i-тите най-малки елементи на масива.

След изпълнение на вътрешния цикъл $a[i]$ съдържа $\min\{a[i], \dots, a[n-1]\}$.

Освен това $a[i] \geq \max\{a[0], \dots, a[i-1]\}$, така че при следващата итерация на цикъла имаме:

$i = k$,

елементите $a[0], \dots, a[i]$ са сортирани и в тях се съдържат $i + 1$-вите най-малки елементи на масива - твърдението отново е вярно.

Терминация: При последната итерация имаме:

$i = n - 1$,

елементите $a[0], \dots, a[n-2]$ са сортирани и в тях се съдържат $n-1$-вите най-малки елементи на масива, от което следва, че $a[n-1]$ съдържа най-големия елемент на масива, така че целият масив $a[0], \dots, a[n-1]$ е сортиран.

Тъй като Selection Sort използва единствено операция `swap()` за промяна съдържанието на масива, то след изпълнение на алгоритъма елементите няма да се променят, а само ще бъдат разместени. Този факт, заедно с горните две инварианти, доказва че Selection Sort е коректен сортиращ алгоритъм.

Пример 4: Даденият алгоритъм връща x^n:

```
int exp_by_sqr(int x, int n)
{
    if (n == 0) return 1;
    if (n == 1) return x;
    if (n%2 == 0) return exp_by_sqr(x*x, n/2);
    return x*exp_by_sqr(x*x, (n - 1)/2);
}
```

Твърдение: Алгоритъмът връща x^n.

База:

При $n = 0$ алгоритъмът връща 1, което е x^0 - вярно

При $n = 1$ алгоритъмът връща x, което е x^1 - вярно

Стъпка:

Нека $n > 1$ е четно и алгоритъмът е коректен за $\frac{n}{2}$ - връща $x^{\frac{n}{2}}$

Тогава той връща $(x^2)^{\frac{n}{2}} = x^n$.

Нека $n > 1$ е нечетно и алгоритъмът е коректен за $\frac{n-1}{2}$ - връща $x^{\frac{n-1}{2}}$

Тогава той връща $x.(x^2)^{\frac{n-1}{2}} = x^n$.

Във всички случай алгоритъмът връща x^n.