Niezawodne Systemy Informatyczne

Lista 2

PRZEMYSŁAW KOBYLAŃSKI

Rozwiąż samodzielnie poniższe zadania i przedstaw prowadzącemu najpóźniej na 9. zajęciach.

Przy każdym z zadań podano maksymalną do uzyskania liczbę punktów.

Za zadanie otrzymuje się:
- 100% punktów gdy program **gnatprove** udowodnił wszystkie asercje (*Assert*), kontrakty (*Post*), kontrole indeksów (*index check*) i zakresów (*overflow check*) oraz nie użyto założeń (*Assume*);
- 75% punktów gdy program **gnatprove** udowodnił wszystkie asercje, kontrakty, kontrole indeksów;
- 50% punktów gdy program **gnatprove** udowodnił wszystkie kontrakty i kontrole indeksów (nieudowodniona asercja *Assert* będzie uznawana za użycie założenia *Assume* i wymaga uzasadnienia słownego podczas zaliczania).

Każde użyte założenie *Assume* należy udowodnić prowadzącemu podczas zaliczania listy ale zasadność jego użycia może być zakwestionowana przez prowadzącego a ma on głos rozstrzygający.

Wskazówka: poczytaj o parametrach polecenia **gnatprove** (*level*, *steps*, *timeout* i innych).

Zadanie 1 (16 pkt)

Poniżej zamieszczono fragment funkcji **IsPrime**, która sprawdza czy jej argument **N** jest liczbą pierwszą.

```
function IsPrime (N : in Positive) return Boolean
  with
    SPARK_Mode,
    Pre => N >= 2,
    Post => (if IsPrime'Result then
                (for all I in 2 .. N - 1 => N rem I /= 0)
             else
                (for some I in 2 .. N - 1 => N rem I = 0))
is
begin
   …
end IsPrime;
```

Napisz funkcję **IsPrime** i udowodnij jej poprawność programem **gnatprove**.

Napisz procedurę **Main**, która testuje poprawność działania funkcji **IsPrime**.

Zadanie 2 (8 pkt)

Poniżej przedstawiono specyfikację pakietu **Max2**, który zawiera funkcję **FindMax2** znajdującą drugą co do wielkości wartość w tabliczy całkowitych liczb dodatnich (tzn. wszystkie poza największą są od niej mniejsze lub równe). Jeśli nie istnieje druga co do wielkości wartość (wszystkie są sobie równe), to funkcja **FindMax2** powinna zwrócić wartość 0 (odpowiednik minus nieskończoności dla typu **Positive**).

```
package Max2 with SPARK_Mode is

   type Vector is array (Integer range <>) of Positive;

   function FindMax2 (V : Vector) return Integer
     with
        Pre => V'First < Integer'Last and V'Length > 0,
        Post => FindMax2'Result >= 0 and
                (FindMax2'Result = 0 or
                  (for some I in V'Range => FindMax2'Result = V(I))) and
                (if FindMax2'Result /= 0 then
                   (for some I in V'Range => V(I) > FindMax2'Result)) and
                (if FindMax2'Result = 0
                 then
                       (for all I in V'Range =>
                         (for all J in V'Range => V(I) = V(J)))
                 else
                       (for all I in V'Range =>
                         (if V(I) > FindMax2'Result then
                           (for all J in V'Range => V(J) <= V(I)))));

end Max2;
```

Napisz treść pakietu **Max2** tak aby program **gnatprove** udowodnił, że funkcja **FindMax2** poprawnie znajduje drugą co do wielkości liczbę w tablicy.

Napisz procedurę **Main**, która testuje poprawność działania funkcji **FindMax2**.

Zadanie 3 (4 pkt)

Poniżej przedstawiono specyfikację funkcji **Sqrt**, która wylicza pierwiastek kwadratowy danej liczby zmiennopozycyjnej **X** z zadaną względną dokładnością **Tolerance**, tż.

$$\frac{|X - Sqrt'Result^2|}{X} \leq Tolerance$$

```
function Sqrt (X : Float; Tolerance : Float) return Float
  with
    SPARK_Mode,
    Pre => X >= 0.0 and X <= 1.8E19 and
           Tolerance > Float'Model_Epsilon and Tolerance <= 1.0,
    Post => abs (X - Sqrt'Result ** 2) <= X * Tolerance
is
   …
begin
   …
```

```
end Sqrt;
```

Napisz treść funkcji **Sqrt** tak aby program **gnatprove** udowodnił poprawność jej działania.

Napisz procedurę **Main**, która testuje poprawność działania funkcji **Sqrt**.