Билет № 1

Задача 1.

Дана implicit спецификация функции, описать функцию эквивалентную данной в explicit форме. Специфицировать слабейшие предусловия.

```
value
        f : Real >< Real  -~-> write x Real
        f(a,b) as y
                post    if a>b then ( x / y = a ) /\ ((x↑2) + y = b ) else y = x /\ a / x = x' end
```

Решение:

```
value
        f : Real >< Real  -~-> write x Real
        f(a,b) is
                if a > b then
                        if a~=0.0 then
                        x:= ((0.0-1.0 + sqrt(1+4*a*b) ) |^| (0.0-1.0 - sqrt(1+4*a*b) )) / (2.0*a); x / a
                                else x:=0.0; b
                        else x:=a/x
                end
        pre     ((a>b) => (1+4*a*b >= 0.0)) /\
                        ((a<=b) => (x~= 0.0))
```

Задача 2.

Определить типы данных, дать явную спецификацию функций, определить предусловия частично-вычислимых функций.

```
type S, E
value
create : Unit -> S,
push_down : E >< S -> S,
push_right : E >< S -> S,
pop_up : S -~-> S,
```

```
pop_left : S  -~-> S,
get : S -~-> E,
axiom
        all e : E, s : S :-
        pop_up( push_down( e, s ) ) is s,
        all e : E, s : S :-
        pop_up( push_right( e, s ) ) is pop_up( s ),
        all e : E, s : S :-
        get( push_down( e, s ) ) is e,
        all e : E, s : S :-
        get( push_right( e, s ) ) is e,
        all e : E, s : S :-
        pop_left( push_right( e, s ) ) is s
```

Решение:

```
type S = L-list, L = E-list, E
value
create : Unit -> S
create () is <..>,
push_down : E >< S -> S
push_down (e,s) is      <. <.e.> .>^s,

push_right : E >< S -> S
push_right (e, s) is    if s = <...> then <. <. e.> .>
                        else <. <.e.> ^ hd s .> ^ tl s
                        end,
pop_up : S -~-> S
pop_up (s) is tl s
pre s ~= <..>,

pop_left : S  -~-> S
pop_left (s) is <. tl hd s .> ^tl s
pre s ~= <..> /\ hd s ~= <..>
(pre s ~= <..> /\ len hd s > 1) – тоже верно,

get : S -~-> E
get (s) is hd hd s
pre s ~= <..> /\ hd s ~= <..>
```

Билет № 2

Задача 1.

Дана explicit спецификация функции, описать ее в implicit форме.

```
value
        f : Int >< Int >< Int -> write x Int >< Int
        f(a,b,c) is if a=b then
                (if a+b > c then c else a+b end, a*b*(if c>0 then x:=c; c else 0-c
                end))
                else (a+b, x:=b; x-b)
                end
```

Решение:

```
value
        f : Int >< Int >< Int -> write x Int >< Int
        f(a,b,c) as  (v, w)
        post
                if a=b then
                        if a+b > c then v=c else  v=a+b end /\
                        if c>0 then x=c /\  w= a*b*c else x=x' /\ w= a*b*(0-c) end
                else v=a+b /\ x=b /\ w=a-b
                end
```

Задача 2.

Разработать спецификацию группы функций.

Определить типы данных и дать explicit определение функций add и del, удовлетворяющих следующим аксиомам. Если необходимо, специфицировать предусловия.

```
type S, K, V
value
create : Unit -> S,
add : K >< V >< S -> S,
del : K >< S -~-> S,
get : K >< S -~-> V
axiom
        all k1,k2 : K, v2 : V, s : S :-
        get (k1, add (k2, v2, s)) is        if k1 = k2 then v2
                                            else get( k1, s )
                                            end,
        all k1,k2 : K, v2 : V, s : S :-
        del( k1, add( k2, v2, s )) is       if k1 = k2 then del (k1, s)
                                            else add(k2, v2, del( k1, s ) )
                                            end
```

Решение:

```
type S = K –m-> V, K, V
value
create : Unit -> S
create () is [ ],
add : K >< V >< S -> S
add (k,v,s) is
        s !! [k +> v],
del : K >< S -~-> S
del (k, s) is
        s \ {k}
        pre true
```

Билет № 3

Задача 1.

Дано explicit определение функции. Написать implicit-спецификацию функции эквивалентной данной.

```
value
        f : Int ><Int >< Int -> write x, y Int >< Int
        f(a,b,c) is if a=b then
                        x:=y; (if a+b > c then c else a+b end,
                        y:=c; b*(if c>0 then x:=c; c else 0-c end))
                else (y:=a+b; y, a-b) end
```

Решение:

```
value
        f : Int ><Int >< Int -> write x Int >< Int
        f(a,b,c) as  (v, w)
        post
                if a=b then
                        y=c /\
                        if c > 0 then x=c else x=y' end /\
                        if a+b > c then v=c else  v=a+b end /\
                        if c>0 then w= b*c else w= b*(0-c) end
                else y=a+b /\ v=a+b /\ x=x'  /\ w=a-b
                end
```

Задача 2.

Разработать явную спецификацию группы функций, удовлетворяющих следующим аксиомам.

Если необходимо, специфицировать предусловия.

```
type S, E
value
create : Unit -> S,
down : E >< S -> S,
right : E >< S -> S,
up : S -~-> S,
left : S  -~-> S,
get : S -~-> E
axiom
        all e : E, s : S :-
        get(right( e, s ) ) is e,
        all e : E, s : S :-
        left(right( e, s ) ) is s,
        all e : E, s : S :-
        up(down( e, s ) ) is s,
        all e : E, s : S :-
        up(right( e, s ) ) is up( s ),
        all e : E, s : S :-
        get(down( e, s ) ) is e
```

Решение:

```
type S = L-list, L = E-list, E
value
create : Unit -> S
create () is <..>,
down : E >< S -> S
down (e,s) is     <. <.e.> .>^s,
right : E >< S -> S
right (e, s) is     if s = <...> then <. <. e.> .>
                              else <. <.e.> ^ hd s .> ^ tl s
                              end,
up : S -~-> S
up (s) is tl s
pre s ~= <..>,
left : S  -~-> S
left (s) is <. tl hd s .> ^tl s
pre s ~= <..> /\ hd s ~= <..>,
get : S -~-> E
get (s) is hd hd s
pre s ~= <..> /\ hd s ~= <..>
```

Билет № 4

Задача 1.

Дана explicit спецификация функции, описать ее в implicit форме.

```
value
        f : Int >< Int >< Int -> write x Int >< Int
        f(a,b,c) is
                if x>0  then x:=x * 2 end;
                if a=b then
                (if a+b > c then c else a+b end, a*b*(if c>0 then x:=c; c else 0-c
                end))
                else (a+b, x:=b; a-b)
                end
```

Решение:

```
value
        f : Int >< Int >< Int -> write x Int >< Int
        f(a,b,c) as  (v, w)
        post
                if x'>0 then
                        if b then x = x' * 2
                        else if c>0 then x=c else x=x'*2 end
                        end
                else x=b
                end /\
                if a=b then
                        if a+b > c then v=c else  v=a+b end /\
                        if c>0 then w= a*b*c else w= a*b*(0-c) end
                else v=a+b /\ w=a-b
                end
```

Задача 2.[1]

Доказать, что вторая спецификация является (или не является) уточнением первой.

```
value
f1  : A >< L -~ -> L
f2  : A >< L -~ -> L
f3  : L -> Nat
```

[1] Условия задачи исправлены.

axiom

all a : A, b : L :-
f3(f1(a,b)) **is** f3(b),
all a : A, b : L :-
f3(f2(a,b)) **is** 2*f3(b),
all a: A, b : L :-
f1(a,b) **is** f2(a,b)
pre f3(b) = 1

---

type

L = A**-list**

value

f1 : A >< L -~ -> L
f1(a, b) **is** <. a .> ^ **tl** b
pre b ~= <..>,

f2 : A >< L -~ -> L
f2(a, b) **is tl** b ^ <. a .> ^ b
pre b ~= <..>,

f3 : L -> **Nat**
f3(l) **is len** l

Решение

Рассмотрим аксиому 3

[[**all** a: A, b : L :- f1(a,b) **is** f2(a,b) **pre** f3(b) = 1]] all_introduction

[[f1(a,b) **is** f2(a,b) **pre** f3(b) = 1]] unfold f3

[[**if len** b = 1 **then** f1(a,b) **is** f2(a,b) **else true end is true**]] unfold f1

[[**if len** b = 1 **then** [[f1(a, b) **is** <. a .> ^ **tl** b **is** f2(a,b) **else true end is true**]] **tl** from <.e.> = <..>

[[**if len** b = 1 **then** [[f1(a, b) **is** <. a .> ^ <..> **is** f2(a,b) **else true end is true**]] … ^<..> = …

[[**if len** b = 1 **then** [[f1(a, b) **is** <. a .> **is** f2(a,b) **else true end is true**]] unfold f2

[[**if len** b = 1 **then** [[f1(a, b) **is** <. a .> **is** f2(a, b) **is tl** b^ <. a .> ^b **else true end is true**]]
<..>…=…
[[**if len** b = 1 **then** [[f1(a, b) **is** <. a .> **is** f2(a, b) **is** <. a .> ^b **else true end is true**]]
(**len** l >0 /\ <.e.>^l~=<.e.>) **is false**

false is true is_annihilation

false

Ответ: Вторая спецификаций не является уточнением первой

[[f1(a, b) **is** <. a .> ^ **tl** b **is** f2(a,b) **pre** f3(b) = 1]] unfold f1

Билет № 5

Задача 1.

Дана explicit спецификация функции, описать функцию эквивалентную данной в implicit форме.

```
value
        f : Int >< Int >< Int -> write x Int >< Int
        f(a,b,c) is if a=b then
                (if a+b > c then c else a+b end,
                 a*b*(if c>0 then x:=c; c else 0-c end))
                else (a+b, a-b)
                end
```

Решение:

```
value
        f : Int >< Int >< Int -> write x Int >< Int
        f(a,b,c) as  (v, w)
        post
                if a=b then
                        if a+b > c then v=c else  v=a+b end /\
                        if c>0 then x=c /\  w= a*b*c else x=x' /\ w= a*b*(0-c) end
                else v=a+b /\ x=b /\ w=a-b
                end
```

Задача 2.

Доказать, что вторая спецификация является (или не является) уточнением первой.

```
value
f1  : A >< L -~ -> L
f2  : A >< L -~ -> L
f3  : L -> Nat
axiom
        [ first ]
        all a : A, b : L :-
        f3(f1(a,b)) is f3(b) + 1,
        [second ]
        all a : A, b : L :-
        f3(f2(a,b)) is f3(b) + 1,
        [ third ]
        all a: A, b : L :-
        f1(a,b) is f2(a,b)
        pre f3(a^b) <= 1
```

type

L = A**-list**

value

f1 : A >< L -~ -> L

f1(a, b) **is** <. a .> ^ b,

f2 : A >< L -~ -> L

f2(a, b) **is** b ^ <. a .> ,

f3 : L -> **Nat**

f3(l) **is card elems** l

Доказательство:

[first]

[[**all** a : A, b : L :- f3(f1(a,b)) **is** f3(b) + 1]] all_introduction

[[f3(f1(a,b)) **is** f3(b) + 1]] unfold f3

[[**card elems** f1(a,b) **is card elems** b + 1]] unfold f1

[[**card elems** <. a .> ^ b **is card elems** b + 1]] unfold **elems**

[[**card if** a **isin elems** b **then elems** b **else elems** { a } **union** b **end is card elems** b + 1]]

unfold c**ard**

[[**if** a **isin elems** b **then card elems** b **else card (elems** { a } **union** b) **end is card elems** b + 1]]

unfold **if**

[[**if** a **isin elems** b **then card elems** b **else card (elems** { a } **union** b) **end is card elems** b + 1]]

calculate **card**

[[**if** a **isin elems** b **then card elems** b **else card elems** b + 1 **end is card elems** b + 1]]

calculate **if**

[[**if** a **isin elems** b **then false else true end is true**]]

is_annihilation

[[**false**]]

Ответ: Вторая спецификация не является уточнением первой.