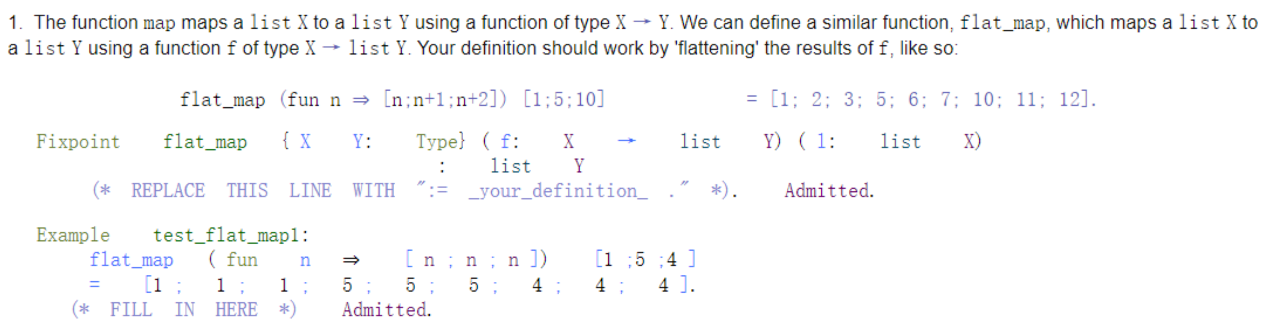
函数语言程序设计作业8

10205101530-赵晗瑜



1. The function map maps a list X to a list Y using a function of type X → Y. We can define a similar function, flat_map, which maps a list X to a list Y using a function f of type X → list Y. Your definition should work by 'flattening' the results of f, like so:

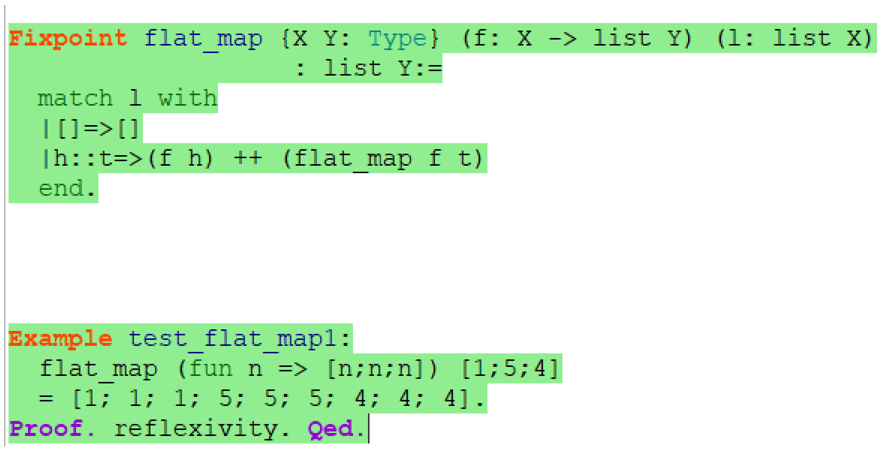
```
        flat_map (fun n ⇒ [n;n+1;n+2]) [1;5;10]         = [1; 2; 3; 5; 6; 7; 10; 11; 12].
Fixpoint flat_map {X Y: Type} (f: X → list Y) (l: list X)
                   : list Y
  (* REPLACE THIS LINE WITH ":= _your_definition_ ." *). Admitted.

Example test_flat_map1:
  flat_map (fun n ⇒ [n;n;n]) [1;5;4]
  = [1; 1; 1; 5; 5; 5; 4; 4; 4].
 (* FILL IN HERE *) Admitted.
```

[ans]

```
Fixpoint flat_map {X Y: Type} (f: X -> list Y) (l: list X)
                   : list Y:=
  match l with
  |[]=>[]
  |h::t=>(f h) ++ (flat_map f t)
  end.

Example test_flat_map1:
  flat_map (fun n => [n;n;n]) [1;5;4]
  = [1; 1; 1; 5; 5; 5; 4; 4; 4].
Proof. reflexivity. Qed.
```

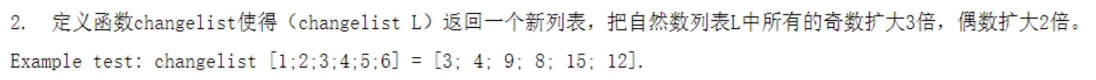


```
Fixpoint flat_map {X Y: Type} (f: X -> list Y) (l: list X)
                   : list Y:=
  match l with
  |[]=>[]
  |h::t=>(f h) ++ (flat_map f t)
  end.



Example test_flat_map1:
  flat_map (fun n => [n;n;n]) [1;5;4]
  = [1; 1; 1; 5; 5; 5; 4; 4; 4].
Proof. reflexivity. Qed.
```

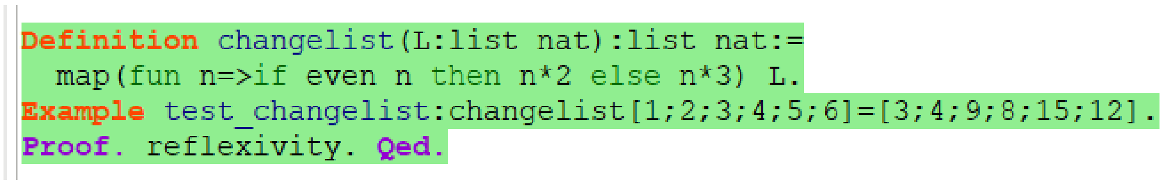
[运行结果]


2. 定义函数changelist使得（changelist L）返回一个新列表，把自然数列表L中所有的奇数扩大3倍，偶数扩大2倍。

Example test: changelist [1;2;3;4;5;6] = [3; 4; 9; 8; 15; 12].

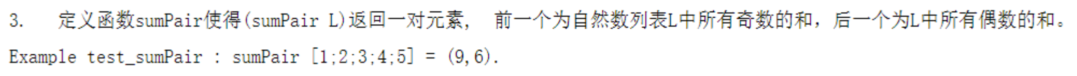
[ans]

```
Definition changelist(L:list nat):list nat:=
  map(fun n=>if even n then n*2 else n*3) L.
Example test_changelist:changelist[1;2;3;4;5;6]=[3;4;9;8;15;12].
Proof. reflexivity. Qed.
```



```
Definition changelist(L:list nat):list nat:=
  map(fun n=>if even n then n*2 else n*3) L.
Example test_changelist:changelist[1;2;3;4;5;6]=[3;4;9;8;15;12].
Proof. reflexivity. Qed.
```

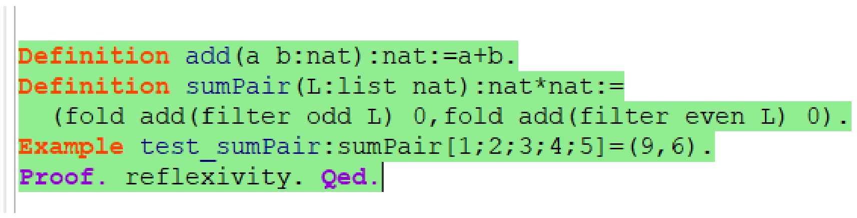
[运行结果]


3. 定义函数sumPair使得(sumPair L)返回一对元素，前一个为自然数列表L中所有奇数的和，后一个为L中所有偶数的和。

Example test_sumPair : sumPair [1;2;3;4;5] = (9,6).

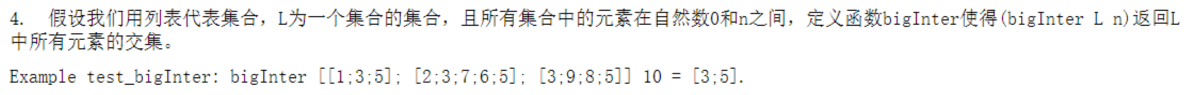
[ans]

```
Definition add(a b:nat):nat:=a+b.
Definition sumPair(L:list nat):nat*nat:=
  (fold add(filter odd L) 0,fold add(filter even L) 0).
Example test_sumPair:sumPair[1;2;3;4;5]=(9,6).
Proof. reflexivity. Qed.
```



```
Definition add(a b:nat):nat:=a+b.
Definition sumPair(L:list nat):nat*nat:=
  (fold add(filter odd L) 0,fold add(filter even L) 0).
Example test_sumPair:sumPair[1;2;3;4;5]=(9,6).
Proof. reflexivity. Qed.
```

[运行结果]


4. 假设我们用列表代表集合，L为一个集合的集合，且所有集合中的元素在自然数0和n之间，定义函数bigInter使得(bigInter L n)返回L中所有元素的交集。

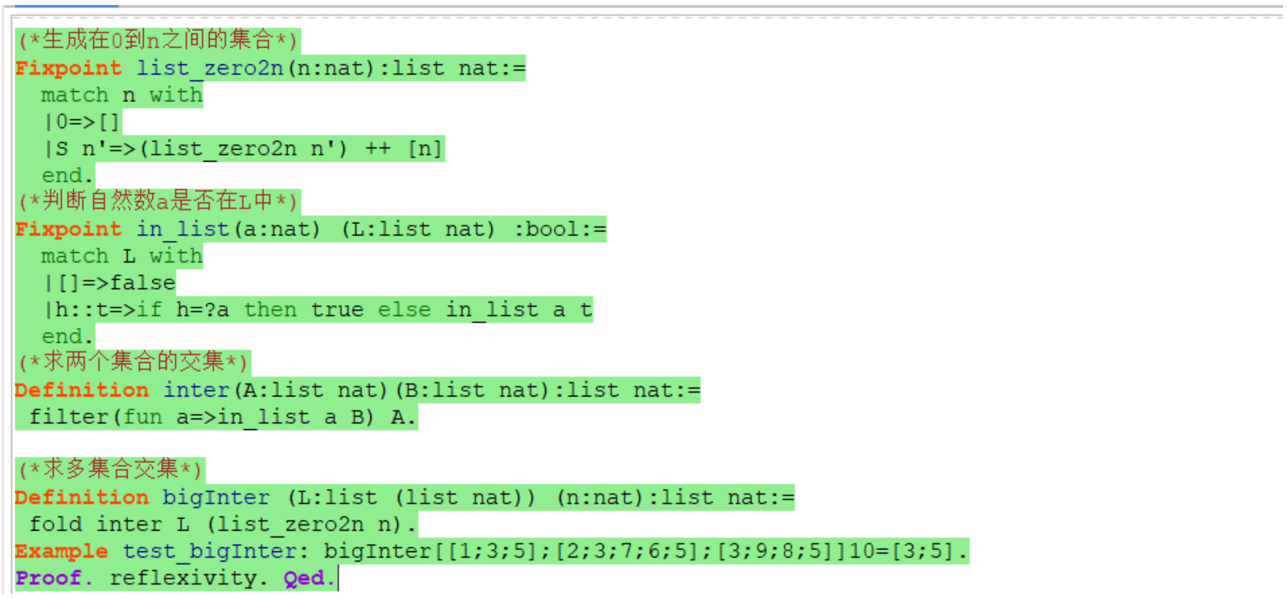
Example test_bigInter: bigInter [[1;3;5]; [2;3;7;6;5]; [3;9;8;5]] 10 = [3;5].

[ans]

```
(*生成在0到n之间的集合*)
Fixpoint list_zero2n(n:nat):list nat:=
  match n with
  |O=>[]
  |S n'=>(list_zero2n n') ++ [n]
  end.
(*判断自然数a是否在L中*)
Fixpoint in_list(a:nat) (L:list nat) :bool:=
  match L with
  |[]=>false
  |h::t=>if h=?a then true else in_list a t
  end.
(*求两个集合的交集*)
Definition inter(A:list nat)(B:list nat):list nat:=
 filter(fun a=>in_list a B) A.

(*求多集合交集*)
Definition bigInter (L:list (list nat)) (n:nat):list nat:=
 fold inter L (list_zero2n n).
Example test_bigInter: bigInter[[1;3;5];[2;3;7;6;5];[3;9;8;5]]10=
[3;5].
Proof. reflexivity. Qed.
```

[运行结果]



```
(*生成在0到n之间的集合*)
Fixpoint list_zero2n(n:nat):list nat:=
  match n with
  |O=>[]
  |S n'=>(list_zero2n n') ++ [n]
  end.
(*判断自然数a是否在L中*)
Fixpoint in_list(a:nat) (L:list nat) :bool:=
  match L with
  |[]=>false
  |h::t=>if h=?a then true else in_list a t
  end.
(*求两个集合的交集*)
Definition inter(A:list nat)(B:list nat):list nat:=
 filter(fun a=>in_list a B) A.

(*求多集合交集*)
Definition bigInter (L:list (list nat)) (n:nat):list nat:=
 fold inter L (list_zero2n n).
Example test_bigInter: bigInter[[1;3;5];[2;3;7;6;5];[3;9;8;5]]10=[3;5].
Proof. reflexivity. Qed.
```