浙江大学城市学院实验报告

课程名称 跨平台脚本开发技术

实验项目名称 实验五 函数式编程

学生姓名 吴成洋 专业班级 软件工程 1404 学号 31401417

实验成绩 _____ 指导老师（签名 ）_____ 日期 _____

注意：

- 务请保存好各自的源代码，已备后用。
- 请把作业保存为 pdf 上传到 BB 平台，请务必在截止日期前提交。

实验目的：

掌握函数式编程概念、原理、实现。

实验内容：

1.实现 fibonacci 数列 1 1 2 3 5 ....

测试对比采用/没有采用 memoize 技术运行时间 fibo(10) fibo(100),fibo(1000),fibo(10000) 50,500,5000,50000 次所需要的时间.

```
var memoize = function(f) {
  var cache = {};
  return function() {
      //JSON 是 JS 提供的一个工具对象 (Utility)
      var arg_str = JSON.stringify(arguments);
      cache[arg_str] = cache[arg_str] || f.apply(f, arguments);
      return cache[arg_str];
  };
  };
//上述 memoize 函数可以直接用 ramda 的库函数 R.memoize
function fibo(n){
// your code
}
```

```
function testfibo(n,times){

    //your code

}

//Date.now()   可以返回当前毫秒数
```

2.柯里化函数练习

```
完成下面的代码，练习柯里化函数 curry function

var R = require('ramda');

// 练习 1

//==============

// 重构使之成为一个 curry 函数

var words = function(str) {

  return split(' ', str);

};

// 练习 1a

//==============

// 使用 map 创建一个新函数，使之能够操作字符串数组

var sentences = undefined;

// 练习 2

//==============

// 重构使之成为一个 curry 函数

var filterQs = function(xs) {

  return filter(function(x){ return match(/q/i, x);  }, xs);

};


// 练习 3

//==============

// 使用帮助函数 _keepHighest 重构 max 使之成为 curry 函数

// 无须改动:
```

```
var _keepHighest = function(x,y){ return x >= y ? x : y; };

// 重构这段代码:

var max = function(xs) {

  return reduce(function(acc, x){

    return _keepHighest(acc, x);

  }, -Infinity, xs);

};

// 彩蛋 1:

// ============

// wrap array's slice to be functional and curried.

// 包裹数组的 slice 函数使之成为 curry 函数

// //[1,2,3].slice(0, 2)

var slice = undefined;


// 彩蛋 2:

// ============

// 借助  slice 定义一个 take  curry 函数，接受 n 个元素为参数。


var take = undefined;
```

3. 函数组合练习

```
// 在当前目录 npm install ramda accounting
// require('../../support');

var R = require('ramda');
var accounting = require('accounting');

// 示例数据
var CARS = [
    {name: "Ferrari FF", horsepower: 660, dollar_value: 700000, in
_stock: true},
```

```
    {name: "Spyker C12 Zagato", horsepower: 650, dollar_value: 648
000, in_stock: false},
    {name: "Jaguar XKR-S", horsepower: 550, dollar_value: 132000,
in_stock: false},
    {name: "Audi R8", horsepower: 525, dollar_value: 114200, in_st
ock: false},
    {name: "Aston Martin One-77", horsepower: 750, dollar_value: 1
850000, in_stock: true},
    {name: "Pagani Huayra", horsepower: 700, dollar_value: 130000
0, in_stock: false}
  ];

// 练习 1:
// ============
// 使用 R.compose() 重写下面这个函数。提示：R.prop() 是 curry 函数
var isLastInStock = function(cars) {
  var last_car = R.last(cars);
  return R.prop('in_stock', last_car);
};

// 练习 2:
// ============
// 使用 R.compose()、R.prop() 和 R.head() 获取第一个 car 的 name
var nameOfFirstCar = undefined;


// 练习 3:
// ============
// 使用帮助函数 _average 重构 averageDollarValue 使之成为一个组合
var _average = function(xs) { return R.reduce(add, 0, xs) / xs.len
gth; }; // <- 无须改动

var averageDollarValue = function(cars) {
  var dollar_values = R.map(function(c) { return c.dollar_value;
 }, cars);
  return _average(dollar_values);
};


// 练习 4:
// ============
// 使用 compose 写一个 sanitizeNames() 函数，返回一个下划线连接的小写
字符串：例如：sanitizeNames(["Hello World"]) //=> ["hello_world"]。
```

```
var _underscore = R.replace(/\W+/g, '_'); //<-- 无须改动，并在 sanit
izeNames 中使用它

var sanitizeNames = undefined;


// 彩蛋 1:
// ============
// 使用 compose 重构 availablePrices

var availablePrices = function(cars) {
  var available_cars = R.filter(R.prop('in_stock'), cars);
  return available_cars.map(function(x){
    return accounting.formatMoney(x.dollar_value);
  }).join(', ');
};


// 彩蛋 2:
// ============
// 重构使之成为 pointfree 函数。提示：可以使用 R.flip()

var fastestCar = function(cars) {
  var sorted = R.sortBy(function(car){ return car.horsepower }, ca
rs);
  var fastest = R.last(sorted);
  return fastest.name + ' is the fastest';
};
```

4. pointfree练习：分析以下示例，学习ramda库函数，并学习pointfree编程风格。

示例 1：

```
var R = require('ramda');
var str = 'Lorem ipsum dolor sit amet consectetur adipiscing elit
';

// 以空格分割单词
var splitBySpace = s => s.split(' ');

// 每个单词的长度
var getLength = w => w.length;
```

```
// 词的数组转换成长度的数组
var getLengthArr = arr => R.map(getLength, arr);

// 返回较大的数字
var getBiggerNumber = (a, b) => a > b ? a : b;

// 返回最大的一个数字
var findBiggestNumber = arr => R.reduce(getBiggerNumber, 0, arr);

var getLongestWordLength = R.pipe(
  R.split(' '),
  R.map(R.length),
  R.reduce(R.max, 0)
);

console.log(getLongestWordLength(str));
```

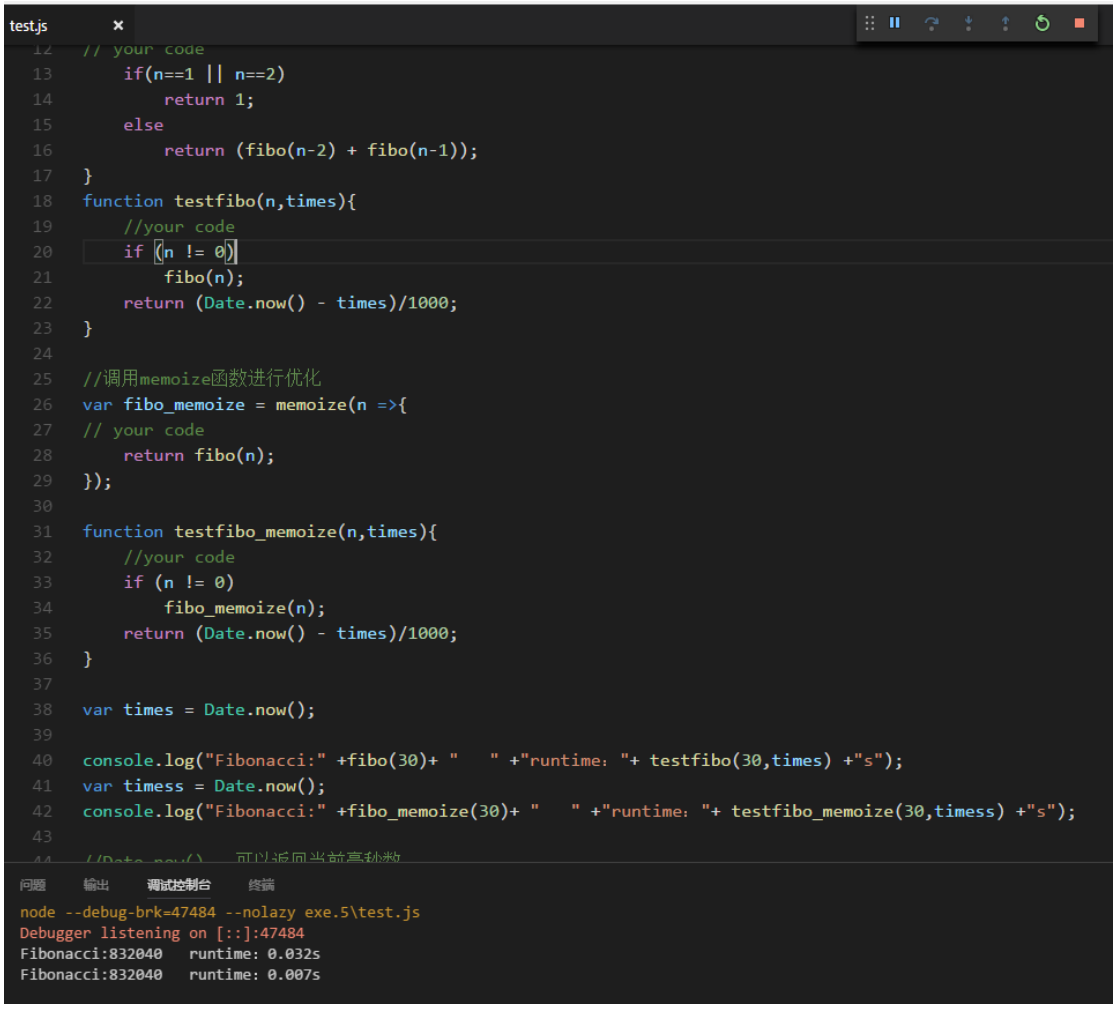
示例 2：

```
var R = require('ramda');
var data = {
    result: "SUCCESS",
    interfaceVersion: "1.0.3",
    requested: "10/17/2013 15:31:20",
    lastUpdated: "10/16/2013 10:52:39",
    tasks: [
        {id: 104, complete: false,           priority: "high",
                  dueDate: "2013-11-29",     username: "Scott",
                  title: "Do something",     created: "9/22/2013"},
        {id: 105, complete: false,           priority: "medium",
                  dueDate: "2013-11-22",     username: "Lena",
                  title: "Do something else", created: "9/22/2013
"},
        {id: 107, complete: true,            priority: "high",
                  dueDate: "2013-11-22",     username: "Mike",
                  title: "Fix the foo",      created: "9/22/2013"},
        {id: 108, complete: false,           priority: "low",
                  dueDate: "2013-11-15",     username: "Punam",
                  title: "Adjust the bar",   created: "9/25/2013"},
        {id: 110, complete: false,           priority: "medium",
                  dueDate: "2013-11-15",     username: "Scott",
                  title: "Rename everything", created: "10/2/2013
"},
```

```
        {id: 112, complete: true,            priority: "high",
                  dueDate: "2013-11-27",     username: "Lena",
                  title: "Alter all quuxes", created: "10/5/2013"}
        // , ...
    ]
};

var fetchData = function () {
  return Promise.resolve(data);
};

// 提取 tasks 属性
var SelectTasks = R.prop('tasks');

// 过滤出指定的用户
var filterMember = member => R.filter(
  R.propEq('username', member)
);

// 排除已经完成的任务
var excludeCompletedTasks = R.reject(R.propEq('complete', true));

// 选取指定属性
var selectFields = R.map(
  R.pick(['id', 'dueDate', 'title', 'priority'])
);

// 按照到期日期排序
var sortByDueDate = R.sortBy(R.prop('dueDate'));

// 合成函数
var getIncompleteTaskSummaries = function(membername) {
  return fetchData().then(
    R.pipe(
      SelectTasks,
      filterMember(membername),
      excludeCompletedTasks,
      selectFields,
      sortByDueDate
    )
  );
};

getIncompleteTaskSummaries('Scott').then(r => console.log(r));
```

实验步骤：

1、



代码如下：

```
var memoize = function(f) {
    var cache = {};
    return function() {
        //JSON 是 JS 提供的一个工具对象 (Utility)
        var arg_str = JSON.stringify(arguments);
        cache[arg_str] = cache[arg_str] || f.apply(f, argument
s);
        return cache[arg_str];
};
};
//上述 memoize 函数可以直接用 ramda 的库函数 R.memoize
function fibo(n){
// your code
```

```
    if(n==1 || n==2)
        return 1;
    else
        return (fibo(n-2) + fibo(n-1));
}
function testfibo(n,times){
    //your code
    if (n != 0)
        fibo(n);
    return (Date.now() - times)/1000;
}

//调用 memoize 函数进行优化
var fibo_memoize = memoize(n =>{
// your code
    return fibo(n);
});

function testfibo_memoize(n,times){
    //your code
    if (n != 0)
        fibo_memoize(n);
    return (Date.now() - times)/1000;
}

var times = Date.now();

console.log("Fibonacci:" +fibo(30)+ "    " +"runtime:
"+ testfibo(30,times) +"s");
var timess = Date.now();
console.log("Fibonacci:" +fibo_memoize(30)+ "    " +"runtime:
"+ testfibo_memoize(30,timess) +"s");

//Date.now()    可以返回当前毫秒数
```

2、

输出端：

test2.js　test3.js　test4.js

```
var E = require('./test3');
console.log(E.words("Jingle bells Batman smells"));
console.log(E.sentences(["Jingle bells Batman smells", "Robin laid an egg"]));
console.log(E.filterQs(['quick', 'camels', 'quarry', 'over', 'quails']));
console.log(E.max([323,523,554,123,5234]));
console.log(E.slice(1)(3)(['a', 'b', 'c']));
console.log(E.take(2)(['a', 'b', 'c']));
```

问题　输出　调试控制台　终端

```
...                     sentences: sentences,
...                     filterQs: filterQs,
...                     max: max,
...                     slice: slice,
...                     take: take
...                   };
{ words: [Function],
  sentences: [Function: f1],
  filterQs: [Function: f1],
  max: [Function: f1],
  slice: [Function],
  take: [Function: f2] }
> var E = require('./test3');
add function f1(a) {
    if (arguments.length === 0 || _isPlaceholder(a)) {
      return f1;
    } else {
      return fn.apply(this, arguments);
    }
  }
undefined
> console.log(E.words("Jingle bells Batman smells"));
[ 'Jingle', 'bells', 'Batman', 'smells' ]
undefined
> console.log(E.sentences(["Jingle bells Batman smells", "Robin laid an egg"]));
[ [ 'Jingle', 'bells', 'Batman', 'smells' ],
  [ 'Robin', 'laid', 'an', 'egg' ] ]
undefined
> console.log(E.filterQs(['quick', 'camels', 'quarry', 'over', 'quails']));
[ 'quick', 'camels', 'quarry', 'over', 'quails' ]
undefined
> console.log(E.max([323,523,554,123,5234]));
5234
undefined
> console.log(E.slice(1)(3)(['a', 'b', 'c']));
[ 'b', 'c' ]
undefined
> console.log(E.take(2)(['a', 'b', 'c']));
[ 'a', 'b' ]
undefined
>
```

练习代码：

```
var R = require('ramda');
// 练习 1
//===============
// 重构使之成为一个 curry 函数
/*var words = function(str) {
  return split(' ', str);
};*/
```

```
//curry 函数
var words = split(' ');


// 练习 1a
//==============
// 使用 map 创建一个新函数，使之能够操作字符串数组
//var sentences = undefined;
//map 函数
var sentences = map(words);

// 练习 2
//==============
// 重构使之成为一个 curry 函数
/*var filterQs = function(xs) {
  return filter(function(x){ return match(/q/i, x);  }, xs);
};*/
//curry 函数
var filterQs = filter(match(/q/i));

// 练习 3
//==============
// 使用帮助函数 _keepHighest 重构 max 使之成为 curry 函数
// 无须改动:
var _keepHighest = function(x,y){ return x >= y ? x : y; };
// 重构这段代码:
/*var max = function(xs) {
  return reduce(function(acc, x){
    return _keepHighest(acc, x);
  }, -Infinity, xs);
};*/
//重构后
var max = reduce(_keepHighest, -Infinity);

// 彩蛋 1:
// ============
// wrap array's slice to be functional and curried.
// 包裹数组的 slice 函数使之成为 curry 函数
// //[1,2,3].slice(0, 2)
//var slice = undefined;
//curry 函数
```

```
var slice = _.curry(function(start, end, xs){ return xs.slice(
start, end); });

// 彩蛋 2:
// ============
// 借助  slice 定义一个 take  curry 函数，接受 n 个元素为参数。
//var take = undefined;
//重构后 curry 函数
var take = slice(0);

module.exports = { words: words,
                   sentences: sentences,
                   filterQs: filterQs,
                   max: max,
                   slice: slice,
                   take: take
                 };
```

3、

```
>
> console.log(isLastInStock(CARS));
false
undefined
> console.log(nameOfFirstCar(CARS));
Ferrari FF
undefined
> console.log(averageDollarValue(CARS));
790700
undefined
> console.log(sanitizeNames(CARS));
[ 'ferrari_ff',
  'spyker_c12_zagato',
  'jaguar_xkr_s',
  'audi_r8',
  'aston_martin_one_77',
  'pagani_huayra' ]
undefined
```

```
> console.log(fastestCar(CARS));
Aston Martin One-77 is the fastest
undefined
```

代码如下

```
// 在当前目录 npm install ramda accounting
require('../../support');
```

```
var R = require('ramda');
var accounting = require('accounting');

// 示例数据
var CARS = [
    {name: "Ferrari FF", horsepower: 660, dollar_value: 700000
, in_stock: true},
    {name: "Spyker C12 Zagato", horsepower: 650, dollar_value:
 648000, in_stock: false},
    {name: "Jaguar XKR-S", horsepower: 550, dollar_value: 1320
00, in_stock: false},
    {name: "Audi R8", horsepower: 525, dollar_value: 114200, i
n_stock: false},
    {name: "Aston Martin One-77", horsepower: 750, dollar_valu
e: 1850000, in_stock: true},
    {name: "Pagani Huayra", horsepower: 700, dollar_value: 130
0000, in_stock: false}
  ];

// 练习 1:
// ============
// 使用 R.compose() 重写下面这个函数。提示：R.prop() 是 curry 函数
/*var isLastInStock = function(cars) {
  var last_car = R.last(cars);
  return R.prop('in_stock', last_car);
};*/

//使用 curry 函数
var isLastInStock = R.compose(R.prop('in_stock'), R.last);

// 练习 2:
// ============
// 使用 R.compose()、R.prop() 和 R.head() 获取第一个 car 的 name
//var nameOfFirstCar = undefined;
//重构
var nameOfFirstCar = R.compose(R.prop('name'),R.head);


// 练习 3:
// ============
// 使用帮助函数 _average 重构 averageDollarValue 使之成为一个组合
```

```
var _average = function(xs) { return R.reduce(R.add, 0, xs) /
xs.length; }; // <- 无须改动

/*var averageDollarValue = function(cars) {
  var dollar_values = R.map(function(c) { return c.dollar_valu
e; }, cars);
  return _average(dollar_values);
};*/
//重构后
var averageDollarValue = R.compose(_average,R.map(R.prop('doll
ar_value')));


// 练习 4:
// ============
// 使用 compose 写一个 sanitizeNames() 函数，返回一个下划线连接的小
写字符串：例如：
sanitizeNames(["Hello World"]) //=> ["hello_world"]。

var _underscore = R.replace(/\W+/g, '_'); //<-- 无须改动，并
在 sanitizeNames 中使用它

//var sanitizeNames = undefined;
//编写 sanitizeNames()函数
var toLowerCase = function(x){ return x.toLowerCase();};
var sanitizeNames = R.map(R.compose(_underscore ,toLowerCase,
R.prop('name')));


// 彩蛋 1:
// ============
// 使用 compose 重构 availablePrices

/*var availablePrices = function(cars) {
  var available_cars = R.filter(R.prop('in_stock'), cars);
  return available_cars.map(function(x){
    return accounting.formatMoney(x.dollar_value);
  }).join(', ');
};*/

//Git 上答案
```

```
var formatPrice = R.compose(accounting.formatMoney(), R.prop("
dollar_value"));
var availablePrices = R.compose( R.join(",") , R.map(formatPri
ce), R.filter( R.prop("in_stock") ) );


// 彩蛋 2:
// ============
// 重构使之成为 pointfree 函数。提示：可以使用 R.flip()

/*var fastestCar = function(cars) {
  var sorted = R.sortBy(function(car){ return car.horsepower }
, cars);
  var fastest = R.last(sorted);
  return fastest.name + ' is the fastest';
};*/

//重构
var append = R.flip(R.concat);
var fastestCar = R.compose(append(' is the fastest'),R.prop('n
ame'),R.last,R.sortBy(R.prop('horsepower')));

module.exports = { CARS: CARS,
                   isLastInStock: isLastInStock,
                   nameOfFirstCar: nameOfFirstCar,
                   fastestCar: fastestCar,
                   averageDollarValue: averageDollarValue,
                   availablePrices: availablePrices,
                   sanitizeNames: sanitizeNames
                 };
```

4、

test8.js

```
var R = require('ramda');
var str = 'Lorem ipsum dolor sit amet consectetur adipiscing elit ';

// 以空格分割单词 var splitBySpace = s => s.split(' ');

// 每个单词的长度 var getLength = w => w.length;


// 词的数组转换成长度的数组 var getLengthArr = arr => R.map(getLength, arr);

// 返回较大的数字 var getBiggerNumber = (a, b) => a > b ? a : b;

// 返回最大的一个数字 var findBiggestNumber = arr => R.reduce(getBiggerNumber, 0, arr);

var getLongestWordLength = R.pipe(   R.split(' '),   R.map(R.length),   R.reduce(R.max, 0) );

console.log(getLongestWordLength(str));
```

```
问题　输出　调试控制台　终端
node --debug-brk=38576 --nolazy exe.5\test8.js
Debugger listening on [::]:38576
11
```

打印出最长字符串的长度

test8.js　test9.js

```
var R = require('ramda');
var data = {
    result: "SUCCESS",
    interfaceVersion: "1.0.3",
    requested: "10/17/2013 15:31:20",
    lastUpdated: "10/16/2013 10:52:39",
    tasks: [
        {id: 104, complete: false,            priority: "high",
                  dueDate: "2013-11-29",      username: "Scott",
                  title: "Do something",      created: "9/22/2013"},
        {id: 105, complete: false,            priority: "medium",
                  dueDate: "2013-11-22",      username: "Lena",
                  title: "Do something else", created: "9/22/2013"},
        {id: 107, complete: true,             priority: "high",
                  dueDate: "2013-11-22",      username: "Mike",
                  title: "Fix the foo",       created: "9/22/2013"},
        {id: 108, complete: false,            priority: "low",
                  dueDate: "2013-11-15",      username: "Punam",
                  title: "Adjust the bar",    created: "9/25/2013"},
        {id: 110, complete: false,            priority: "medium",
```

```
问题　输出　调试控制台　终端
node --debug-brk=3869 --nolazy exe.5\test9.js
Debugger listening on [::]:3869
[ { id: 110,
    dueDate: '2013-11-15',
    title: 'Rename everything',
    priority: 'medium' },
  { id: 104,
    dueDate: '2013-11-29',
    title: 'Do something',
    priority: 'high' } ]
```

打印出 username 是 Scott 且未完成的任务的四个属性，并按日期排序