**全国计算机技术与软件专业技术资格（水平）考试**

**2017年上半年 程序员 下午试卷**

（考试时间 14:00～16:30 共 150 分钟）

|  |
| --- |
| **请按下述要求正确填写答题纸** |

1.在答题纸的指定位置填写你所在的省、自治区、直辖市、计划单列市的名称。

2.在答题纸的指定位置填写准考证号、出生年月日和姓名。

3.答题纸上除填写上述内容外只能写解答。

4.本试卷共 7 道题，试题一至试题四是必答题，试题五至试题六选答 1 道。每

题 15 分，满分 75 分。

5.解答时字迹务必清楚，字迹不清时，将不评分。

6.仿照下面例题，将解答写在答题纸的对应栏内。

**例题**

2017年上半年全国计算机技术与软件专业技术资格（水平）考试日期是（1）

月（2）日。

因为正确的解答是“5 月 20 日”，故在答题纸的对应栏内写上“5”和“20”

（参看下表）。

|  |  |
| --- | --- |
| 例题 | 解答栏 |
| （1） | 5 |
| （2） | 20 |

**试题一（共 20 分）**

阅读下列说明和图，回答问题 1 至问题 3，将解答填入答题纸的对应栏内。

**【说明】**

设有二维整数数组（矩阵）A[1:m,1:n],其每行元素从左至右是递增的，每列元素从上到下是递增的。以下流程图旨在该矩阵中需找与给定整数 X 相等的数。如果找不到则输出“false”;只要找到一个（可能有多个）就输出“True”以及钙元素的下标 i 和 j（注意数组元素的下标从 1 开始）。

例如，在如下矩阵中查找整数 8，则输出伟：True,4,1

2 4 6 9

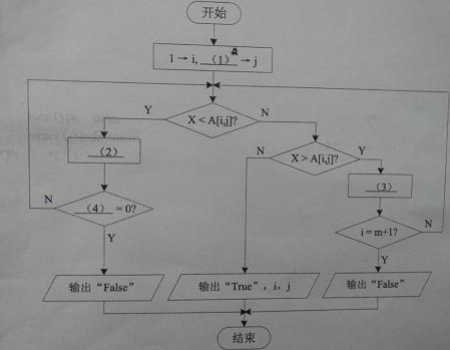
4 5 9 10

6 7 10 12

8 9 11 13

流程图中采用的算法如下：从矩阵的右上角元素开始，按照一定的路线逐个取元素与给定整数 X 进行比较（必要时向左走一步或向下走一步取下一个元素），直到找到相等的数或超出矩阵范围（找不到）。

**【流程图】**



【问题】该算法的时间复杂数是（）

供选择答案：A.O（1） B.O（m+n） C.（m\*n） D,O（m²+n²）

**试题二（共 15 分）**

阅读下列说明和 C 函数，填补函数中的空缺，将解答填入答案纸的对应栏目内。

**【说明】**

函数 isLegal（char\*ipaddr）的功能是判断以点分十进制数表示的 iPV4 地址是否合法。参数 ipadddr 给出表示 iPV4 地址的字符串的首地址，串中仅含数字字符和“.”。若 iPV4 地址合法则返回1，否则反馈 0.判定伟合法的条件是：每个十进制数的值位于整数区间[0,25]，两个相邻的树之间用“.”分隔，共 4 个数、3 个“.”。；例如，192.168.0.15、1.0.0.1 是合法的，192.168.1.256、1.1..1是不合法的。

**【函数】**

int isLegal （char\*ipaddr）

﹛

int flag;

int cur Val; ／／curVal 表示分析出的一个十进制数

int decNum=0,dotNum=0; ／／decNum 用于记录十进制数的个数

／／dotNum 用户记录点的个数

Char\*p=（）

for（;\*p;p++） ﹛

curVal=0;flag=0

While （isdigit（\*p））﹛ ／／判断是否伟数字字符

CurVal=（）+\*p-′0′;

（）

flag=1;

﹜

if（curVal>255）﹛

return 0;

﹜

if （flag）﹛

（）

﹜if（\*p=′.′﹛

dotNum++;

﹜

﹜

if （）﹛

return 1；

﹜

return 0;

﹜

**【试题三】**

阅读下列说明和 C 函数，填补 C 函数中的空缺，将解答填入答案纸的对应栏目内。

**【说明】**

字符串是程序中常见的一种处理对象，在字符串中进行子串的定位、插入和删除是常见的运算。

设存储字符串时不设置结束标志，而是另行说明串的长度，因此串类型定义如下：

Typedef struct ﹛

Char\*str ／／字符串存储空间的起始地址

int lehgth ／／字符串长

int capacity ／／存储空间的容量

﹜SString;

**【函数 1 说明】**

函数 indexStr（S,T,pos）的功能是：在 S 所表示的字符串中，从下标 pos 开始查找 T 所表示字符串首次出现的位置。方法是：第一趟从 S 中下标为 pos、T 中下标伟 0 的字符开始，从左往右逐个对于来比较 S 和 T 的字符，直到遇到不同的字符或者到达 T 的末尾。若到达 T 的末尾，则本趟匹配的起始下标 pos 为 T 出现的位置，结束查找；若遇到了不同的字符，则本趟匹配失效。下一趟从 S 中下标 pos+1 处的字符开始，重复以上过程。若在 S 中找到 T，则返回其首次出现的位置，否则返回-1。

例如，若 S 中的字符串伟″students ents″,T 中的字符串伟″ent″，pos=0，则 T 在 S 中首次出现的位置为 4。

**【C 函数 1】**

int index Str（SString S ,SString T,int pos）

﹛

int i,j:

i （S.length<1||S.length<pos+T.length-1）

return-1;

for（i=pos,j=0;i<S.length &&j<T.length;）﹛

if （S.str[i]==T.str[j]）﹛

i++;j++;

﹜else﹛

i=（）;j=0

﹜

﹜

if （）return i -T.length;

return-1;

﹜

**【函数 2 说明】**

函数 eraseS 位（S，T}的功能是删除字符串 S 中所有与 T 相同的子串，其处理过程为: 首先从字符串 S 的第一个字符（下标为 0）开始查找子串 T，若找到〈得到子串 在 S 中的起始位置），则将串 S中子串 T 之后的所有字符向前移动，将子串 T 覆盖，从而将 其删除，然后重新开始查找下一个子串 T，若找到就用后面的宇符序列进行覆盖，重复上述过程，直到将 S 中所有的子串 T 删除。

例如，若字符串 S 为 “12ab345abab678”、T 为“ab”。第一次找到 "ab" 时（位置为（2），将 "345abab678 "前移，S 中的串改为"12345abab678" ，第二次找到"ab"时（位置为 5）;将 ab678 前移，S 中的串改为 "12345ab678"，第三次找到"ab"时（位置 为 5）;将“678‘前移 ，S 中的串改为 "12345678 "。

**【C 函数 2】**

Void eraseStr（SString\*S,SStringT）

﹛

int i;

int pos;

if （S->length<||T.length<1||S->length<T.length）

return;

Pos=0

for（;;）﹛

／／调用 indexStr 在 S 所表示串的 pos 开始查找 T 的位置

Pos=indexStr（）;

if（pos＝-1） ／／S 所表示串中不存在子串 T

return;

for（i=pos+T.length;i<S->length;i++） ／／通过覆盖来删除自串 T

S->str[（）]=S->str[i];

S->length=（）; ／／更新 S所表示串的长度

﹜

﹜

**试题四（共 15 分）**

阅读以下说明和 C 函数，填补函数中的空缺，将解答填入答题纸的对应栏内。

**【说明】**

简单队列是符合先进先出规则的数据结构，下面用不含有头结点的单向循环链表表示简单队列。

函数 enqueue（queue \*q，KeyType new\_elem） 的功能是将元素new\_elem 加入队尾。

函数 Dnqueue（queue \*q，KeyType \*elem）的功能使将非空队列的队头元素出队（从

队列中删除），并通过参数带回刚出队的元素。

用单向循环链表表示的队列如图 4-1 所示。

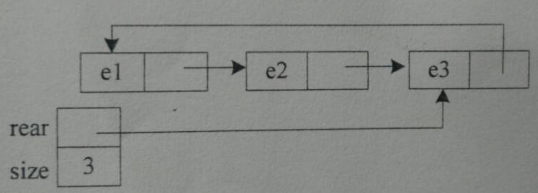


图 4-1 单向循环链表表示的队列示意图

队列及链表结点等相关类型定义如下:

enum {errOr， OK};

typedef int KeyType;

typedef struct qNode﹛

KeyType data;

Struct qNode\*next;

﹜qNode，\*Linkqueue;

Typedef struct﹛

int size;

Link:queue rear;

}queue;

**【C 函数】**

int enqueue（queue\*q,KeyType new\_elem）

﹛ ／／元素 new\_elem 入队列

qNode\*p;

P=（qNode\*）malloc（sizeof（qNode））;

if（!p）

return errOr;

P->data=new\_elem;

if（q->rear）﹛

P->next=q->rear->next;

（）;

﹜

else

P->next=p;

﹙﹚;

q->size++;

return OK;

﹜

int Dequeue（queue\*q,KeyType\*elem）

﹛ ／／出队列

qNode\*p;

if（0＝＝q->size） ／／是空队列

return errOr；

P=（）; ／／令 p 指向队头元素结点

\*elem =p->data;

q->rear->next=（）; ／／将队列元素结点从链表中去除

if（（）） ／／被删除的队头结点是队列中唯一结点

q->rear=NULL ／／变成空队列

free（p）;

q->size--;

return OK;

﹜

**试题五（共 15 分）**

阅读以下说明和 Java 程序，填补代码中的空缺，将解答填入答题纸的对应栏内。

**【说明】**

以下 Jave 代码实现一个简单客户关系管理系统（CrM） 中通过工厂 （Customerrfactory ）对象来创建客户（Customer） 对象的功能。客户分为创建成功的客户 （realCustomer） 和空客户（NullCustomer） 。空客户对象是当不满足特定条件时创建或获取的对象。类间关系如图 5-1 所示。

**【Java 代码】**

Abstract class Customer﹛

Protected String name;

（）boolean isNil（）

（）String getName（）;

﹜

Class realCustomer （）Customer﹛

Public realCustomer（String name ）﹛ return false; ﹜

﹜

Class NullCustomer（）Customer﹛

Public String getName（）﹛ return ″Not Available in Customer Database″; ﹜

Public boolean isNil（）﹛ return true; ﹜

﹜

class Customerfactory {

public String[] names = {"rob"，"Joe"，"Julie"};

public Customer getCustomer（String name） {

for （int i = 0; i < names.length;i++） {

if （names[i].（））﹛

return new realCusωmer（name）;

﹜

﹜

return （）

﹜

﹜

Public class CrM﹛

Public viod get Customer（）﹛

Customerfactory（）

Customer customer1-cf.getCustomer（″rob″）;

Customer customer2=cf.getCustomer（″rob″）;

Customer customer3= cf.getCustomer（″Julie″）;

Customer customer4= cf.getCustomer（″Laura″）;

System.out.println（″customer1.getName（））;

System.out.println（″customer2getName（））;

System.out.println（″customer3.getName（））;

System.out.println（″customer4.getName（））;

﹜

Public static viod main （String[]arge）﹛

CrM crm =new CrM（）;

Crm,getCustomer（）;

﹜

﹜

／\*程序输出为：

Customer

rob

Not Available in Customer Database

Julie

Not Available in Customer Datable

\*／

int main（）﹛

CrM\*crs=newCrM（）;

Crs->getCustomer（）;

Crs->getCustomer（）;

Delete crs;

return（）;

﹜

／\*程序输出为：

Customer

rob

Not Available ini Customer Database

Julie

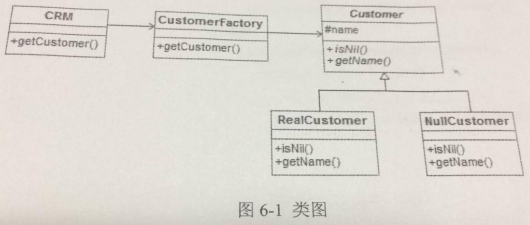
Not Available in Customer Database

**试题六（共 15 分）**

阅读下列说明和 C++代码，填补代码中的空缺，将解答填入答题纸的对应栏内。

**【说明】**

以下 C++代码实现一个简单客户关系管理系统（CrM）中通过工厂（Customerfactory）对象来创建客户（Customer）对象的功能。客户分为创建成功的客户（realCustomer）和空客户（NullCustomer）。空客户对象是当不满足特定条件时创建或获取的对象。类间关系如图6-1 所示。



【C++代码】

#include<iostream>

#include<string>

using namespace std；

class Customer｛

protected:

string name；

public:

（1） boll isNil（）=0；

（2） string getName（）=0；

﹜；

class realCustomer （3）｛

public:

realCustomer（string name）｛this->name=name；﹜

bool isNil（）｛ return false；﹜

string getName（）｛ return name；﹜

﹜；

class NullCustomer （4） ｛

public:

bool isNil（）｛ return true；﹜

string getName（）｛ return 〝Not Available in Customer Database〞； ﹜

﹜；

class Customerfactory｛

public:

string names［3］=｛〝rob〞, 〝Joe〞,〝Julie〞﹜；

public:

Customer\*getCustomer（string name）｛

for （int i=0；i<3；i++）｛

if （names［i］.（5） ）｛

return new realCustomer（name）；

﹜

﹜

return （6）；

﹜

﹜；

class CrM｛

public:

void getCustomer（）｛

Customerfactory\*（7）；

Customer\*customer1=cf->getCustomer（〝rob〞）；

Customer\*customer2=cf->getCustomer（〝Bob〞）；

Customer\*customer3=cf->getCustomer（〝Julie〞）；

Customer\*customer4=cf->getCustomer（〝Laura〞）；

cout<<〝Customers〞<<endl；

cout<<Customer1->getName（） <<endl； delete customer1；

cout<<Customer2->getName（） <<endl； delete customer2；

cout<<Customer3->getName（） <<endl； delete customer3；

cout<<Customer4->getName（） <<endl； delete customer4；

delete cf；

﹜

﹜；

int main（）｛

CrM\*crs=new CrM（）；

crs->getCustomer（）；

delete crs；

return 0；

﹜

/\*程序输出为：

Customers

rob

Not Available in Customer Database

Julie

Not Available in Customer Database

\*/