第一届蓝桥杯预赛试题 c 本科

第一题，以下函数的功能是将任意整数进行因式分解， 请完成该程序的空缺使函数能够正常运算

```
void f(int n)
{
for(int i=2; i&lt;n/2; i++)
{
//————————（1）————————
{
printf("%d ", i);                                    n = n / i;
}
}
if(n&gt;1) printf("%d\n", n);
}
```

第二题，以下函数的功能是将给定字符串进行倒序输出， 请完成该函数空缺部分。

```
char* p = "abcdef";
char* q = (char*)malloc(strlen(p)+1);
for(int i=0, int len=strlen(p); i&lt;len-1; i++)

q__1__ = p[i+1];
q[len-1] = p[0];
——（2）——;
printf("%s\n", q);
```


第三题，

```
int f(int m, int n)
{
int a = 1;
int m1 = m;
//——（3）——
a *= m1–;
int b = 1;
while(n&gt;1)
b *= n–;
return a / b;
}
```

第四题，任意给出一个四位数，
把它重新组成一个四位的最大数和一个最小数，
算出两者间的差。
例如：3721 这个数，可以重组成：7321 和 1237，相数之差为 7321－1237 请

完善下面这个函数，以实现此功能

```
int f(int n)
{
int N[4];

for(int i=0; i&lt;4; i++)
{
N[3-i] = n % 10;
—-（4）—-
}
for(i=0; i&lt;3; i++)
for(int  j=0;  j&lt;3-i;  j++)                                    if(N[j]&gt;N[j+1])
{
int  t  =  N[j+1];                                                N[j+1]  =  N[j];
N[j] = t;                                  }
int n_min=0;
for(i=0; i&lt;4; i++)
n_min = n_min * 10 + N[i] ;              int n_max = 0;
for(i=3; i&gt;=0; i–)
n_max = n_max * 10 + N[i];               return n_max-n_min;
}
```

第五题，假设有 m+n 个人，其中，
m 个人手持面额为 5 角的硬币，
n 个人手持面额为 1 元的硬币，

他们都要乘车买票，
现假设售票员手中无零钞，
票价为 5 角，
下面这个函数就可以算出这 m+n 个人所有可能的买票情况，
请完善此函数。

```
//m: 持有 5 角币的人数
//n: 持有 1 元币的人数
//返回：所有顺利完成购票过程的购票次序的种类数
int f(int m, int n)
{
if(m &lt; n) return 0;
if(n==0) return 1;
return _____（5）_____;
}
```

最后一题：编程题：
注：最后一题的编程题对参赛者的要求相当高，以下代码在你提交自己的程序设计思路前是不可见的。
求二十四点：

```
// Calcu24.cpp : Defines the entry point for the console application. //
```

```
#include “stdafx.h”
#include “conio.h”
#include “stdlib.h”
#include “time.h”
#include “math.h”
#include “string.h”
/*
从一副扑克牌中，任取 4 张。

2-10 按其点数计算(为了表示方便 10 用 T 表示)，J,Q,K,A 统一按 1 计算 要求通过加减乘除四则运算得到数字 24。
本程序可以随机抽取纸牌，并用试探法求解。
*/
void GivePuzzle(char* buf)
{
char card[] =
{ ‘A’ ,’ 2′,’ 3′,’ 4′,’ 5′,’ 6′,’ 7′,’ 8′,’ 9′,’ T’,’ J’,’ Q’,’ K’};
for(int i=0; i&lt;4; i++){
buf = card[rand() % 13];
}
}
void shuffle(char * buf)
{
for(int i=0; i&lt;5; i++){
int k = rand() % 4;
char t = buf[k];
buf[k] = buf[0];
buf[0] = t;
}
}
int GetCardValue(int c)
{
if(c==’ T’) return 10;
if(c&gt;=’ 0′ &amp;&amp; c&lt;=’ 9′) return c –
return 1;
}
char GetOper(int n)
{
switch(n)
{
case 0:
return ‘+’;
case 1:
return ‘-’;
```

```
case 2:
return ‘*’;
case 3:
return ‘/’;
} ’0′;

return ‘ ‘;
}
double MyCalcu(double op1, double op2, int oper)
{
switch(oper)
{
case 0:
return op1 + op2;
case 1:
return op1 – op2;
case 2:
return op1 * op2;
case 3:
if(fabs(op2)&gt;0.0001)
return op1 / op2;
else
return 100000;
}
return 0;
}
void MakeAnswer(char* answer, int type, char* question, int* oper) {
char p[4][3];
for(int i=0; i&lt;4; i++)
{
if( question == ‘T’ )
strcpy(p, “10″);
else
sprintf(p, “%c”, question);
}

switch(type)
{
case 0:
sprintf(answer, “%s %c (%s %c (%s %c %s))”,
p[0], GetOper(oper[0]), p[1], GetOper(oper[1]), p[2], GetOper(oper[2]), p[3]);
break;
case 1:
sprintf(answer, “%s %c ((%s %c %s) %c %s)”,
```

```
p[0], GetOper(oper[0]), p[1], GetOper(oper[1]), p[2], GetOper(oper[2]), p[3]);

break;
case 2:
sprintf(answer, "(%s %c %s) %c (%s %c %s)",
p[0], GetOper(oper[0]), p[1], GetOper(oper[1]), p[2], GetOper(oper[2]), p[3]);
break;
case 3:
sprintf(answer, "((%s %c %s) %c %s) %c %s",
p[0], GetOper(oper[0]), p[1], GetOper(oper[1]), p[2], GetOper(oper[2]), p[3]);
break;
case 4:
sprintf(answer, "(%s %c (%s %c %s)) %c %s",
p[0], GetOper(oper[0]), p[1], GetOper(oper[1]), p[2], GetOper(oper[2]), p[3]);
break;
}
}
bool TestResolve(char* question, int* oper, char* answer)
{
// 等待考生完成
}
return true;
//return false;
}
/*
采用随机试探法：就是通过随机数字产生 加减乘除的 组合，通过大量的测试来命中的解法
提示：
1. 需要考虑用括号控制计算次序的问题 比如：( 10 – 4 ) * ( 3 + A ), 实际上计算次序的数目是有限的：
A*(B*(c*D))
A*((B*C)*D)
(A*B)*(C*D)
((A*B)*C)*D
(A*(B*C))*D
2. 需要考虑计算结果为分数的情况：( 3 + (3 / 7) ) * 7
3. 题目中牌的位置可以任意交换
*/
bool TryResolve(char* question, char* answer)
{
int oper[3];  // 存储运算符，0:加法 1:减法 2:乘法 3:除法


for(int i=0; i&lt;1000 * 1000; i++)
```

```
{
// 打乱纸牌顺序
shuffle(question);

// 随机产生运算符
for(int j=0; j&lt;3; j++)
oper[j] = rand() % 4;
if( TestResolve(question, oper, answer) )    return true;                }
return false;
}
int main(int argc, char* argv[])
{
// 初始化随机种子
srand( (unsigned)time( NULL ) );
char buf1[4];     // 题目
char buf2[30];   // 解答
printf("**************************\n");
printf("计算 24\n");
printf("A J Q K 均按 1 计算，其它按牌点计算\n");
printf("目标是：通过四则运算组合出结果：24\n");
printf("**************************\n\n");
for(;;)
{
GivePuzzle(buf1);   // 出题

printf("题目：");
for(int j=0; j&lt;4; j++){
if(    buf1[j]    ==    '    T    '    )//    、    、    、    初    始    化    buf1[];
printf("10 ");
else
printf("%c ", buf1[j]);
}
printf("\n 按任意键参考答案,,\n");
getch();
if( TryResolve(buf1, buf2) )   // 解题
{

//              //printf("a\n");

//for(int i=0;i&lt;17;i++)
//              printf("%c",buf2);
printf("参考：%s\n", buf2);                              }
else
printf("可能是无解,,\n");
```

```
printf("按任意键出下一题目，x 键退出,,\n");                    if( getch() ==
'x' ) break;
}
return 0;
}
```