# 参考答案

| 得分 | |
|---|---|

一、分析下面程序的缺陷。（本题满分**12**分）

答：**（每行 2 分）**
**（1）比如 把大写 X 改成 x2，符号常量 eps 用大写字母，函数名 cube 首字母改成大写；**
**（2）降低了函数 cube 的独立性（可移植性、通用性）；**
**（3-1）不能；**
**（3-2）函数 cube 的返回值类型为 int，应改为 double；**
**（4-1）分母为 0 的问题；**
**（4-2）循环前增加一个判断：if(!x) return 0;**

| 得分 | |
|---|---|

二、分析程序的计算结果。（本题满分**13**分）**客观题，多输出、顺序错都不得分**

**1.（4分）一个数据 2 分**

```
    int i, j;
    for(i = 1; i <= 50; ++i)
        for(j = 1; j <= 50; ++j)
            if(i * j == 50) break;
    printf("%d, %d\n", i, j);//cout << i << ", " << j << endl;
```

**//该程序片段执行后的输出结果为：**_____51, 1_____

**2.（5分）一个数据 0.5 分**

```
#include <stdio.h>  //#include <iostream>
                    //using namespace std;
void Swap1(int x, int y);
void Swap2(int* x, int* y);
void Swap3(int* x, int* y);
void Swap4(int** x, int** y);

int main()
{
    int a = 0, b = 1;
    int* p = &a, * q = &b;
    printf("%d, %d\n", *p, *q);//cout << *p << ", " << *q << endl;
    Swap1(*p, *q);
    printf("%d, %d\n", *p, *q);//cout << *p << "," << *q << endl;
    Swap2(p, q);
    printf("%d, %d\n", *p, *q);//cout << *p << "," << *q << endl;
    Swap3(p, q);
    printf("%d, %d\n", *p, *q);//cout << *p << "," << *q << endl;
    Swap4(&p, &q);
    printf("%d, %d\n", *p, *q);//cout << *p << "," << *q << endl;
    return 0;
}
```

```
void Swap1(int x, int y)
{
    int temp;
    temp = x; x = y; y = temp;
}
void Swap2(int* x, int* y)
{
    int temp;
    temp = *x; *x = *y; *y = temp;
}
void Swap3(int* x, int* y)
{
    int *temp;
    temp = x; x = y; y = temp;
}
void Swap4(int** x, int** y)
{
    int* t;
    t = *x; *x = *y; *y = t;
}
```

//该程序执行后的输出结果为：

____0, 1_____

____0, 1_____

____1, 0_____

____1, 0_____

____0, 1_____

**3．（4分）一个数据4分**

**//该程序执行后的输出结果为：** _____7_____

| 得分 | |
|---|---|

三、根据程序的功能（见注释），纠正程序中的错误。（请将正确写法写在错误行的**右方**或**下方**注释符之后，不得更改程序的结构，使程序能得到正确结果，本题满分**30**分）**（一个bug3分）**

```
#include <stdio.h>  //#include <iostream>
                    //using namespace std;
const int NUM = 6;
const int NAMESIZE = 20;
int Cmp(const char* src1, const char* src2);
void MCopy(const char* dst, const char* src);
   // void MCopy(char* dst, const char* src);  这是半个bug
char* Cate(char* dst, const char* src1, const char* src2);
int main()
{
    const char pre[NAMESIZE] = "Chi";//词根
    const char* lattr[NUM] = {"na","lo","ne","me","la","lu"};//不同人听到的语音
    char nation[NAMESIZE];
    char brics[NAMESIZE];
    int count;                          // int count = 0;
    for (int i = 0; i <= NUM; ++i) // for (int i = 0; i < NUM; ++i)
    {
        nation = Cate(nation, pre, lattr);//将词根与听到的语音合成为单词 存入 nation
            // nation = Cate(nation, pre, lattr[i]);  或 *(lattr+i)
        MCopy(brics, nation);//修正单词中的元音字母 存入 brics
        if (brics == "China")//判断处理后的单词是否正确，即统计听对辅音 n 的次数
                                        // if (!Cmp(brics, "China"))
            ++count;
    }
    printf("%d\n", count);//cout << count << endl;
    return 0;
}
```

```
int Cmp(const char* src1, const char* src2)
{
    while (*src1 == *src2)
    {
        if (*src1 = '\0')          // if (*src1 == '\0')
            return 0;
        ++src1, ++src2;
    }
    return *src1 - *src2;
}//比较两个字符串的大小，相等返回 0

void MCopy(const char dst[], const char src[])
   // void MCopy(char dst[], const char src[])这是另外半个 bug
{
    int i = 0;
    while (src[i] != '\0' || i < NAMESIZE)
                // while (src[i] != '\0' && i < NAMESIZE)
    {
        dst[i] = src[i];
        if (dst[i] == 'e' || dst[i] == 'o' || dst[i] == 'u')
            dst[i] = 'a';
        ++i;
    }
    return;                        // 去掉
    dst[i] = '\0';
}//修正字符串 src 中的元音字母 存入 dst

char* Cate(char* dst, const char* src1, const char* src2)
{
                                   // char* p = dst;
    while (*src1 != '\0')
        *dst++ = *src1++;
    while (*src2 != '\0')
        *dst++ = *src2++;
    *dst = '\0';
    return dst;                    // return p;
}//将字符串 src1 和 src2 合成一个字符串 存入 dst
```

| 得分 | |
|---|---|

四、可使用的库函数包括： malloc、free和输入/输出库函数，可使用的操作包括：new、delete和输入/输出相关的操作。

1. 17 分，每个小 bug 扣 1 分，大 bug 按下方对应功能扣分

```
int getIndexBestArea(int grid[][SIZE])
{
    int maxAreaValue = -1;                          占 1 分
    int maxAreaIndex = 0;                           占 1 分
    for (int i = 0; i <= SIZE - 3; ++i)             起点终点各占 1 分
    {
        for (int j = 0; j <= SIZE - 3; ++j)         起点终点各占 1 分
        {
            int areaValue = 0;                      占 1 分
            for (int k = 0; k < 3; ++k)             起点终点各占 1 分
                for (int t = 0; t < 3; ++t)         起点终点各占 1 分
                    areaValue += grid[i+k][j+t];    行列计算各占 1 分
```

```
            if (areaValue > maxAreaValue)              占 1 分
            {
                maxAreaValue = areaValue;              占 1 分
                maxAreaIndex = SIZE * (i+1) + j+1;     行列计算各占 1 分
            }
        }
    }
    return maxAreaIndex;
}
```

2．28 分，每个小 bug 扣 1 分，大 bug 按下方对应功能扣分

```
#include <stdio.h>  //#include <iostream>
                    //using namespace std;
#include <stdlib>   //
struct Node                              类型构造占1分
{
    int id;         //客户ID
    int points;     //客户的积分
    Node* next;
};
void PrintList(Node* head);
Node* DeleteList(Node* head);
Node* DescCreate(Node* h, Node* p);

int main()                               main占7分
{
    Node* head = NULL;
    int n;
    scanf("%d", &n);//cin >> n;
    int count = 0;
    while(n != 0)
    {
        if (n == -1)                      （在其中占2分）
        {
            head = DeleteList(head);
            PrintList(head);
        }
        else                              （在其中占5分）
        {
            Node* p = (Node*)malloc(sizeof(Node))
                                          //Node* p = new Node;
            p->id = ++count;
            p->points = n;
            p->next = NULL;
            head = DescCreate(head, p);
            PrintList(head);
        }
        scanf("%d", &n);//cin >> n;
    }
    return 0;
}
```

```
void PrintList(Node* head)          占4分
{
    while(head)
    {
        printf("%d:%d ", head->id, head->points);
        //cout << head->id << ":" << head->points << " ";
        head = head->next;
    }
    printf("\n");//cout << "\n";
}

Node* DeleteList(Node* head)        占4分
{
    if (!head)
        return head;
    Node* current = head;
    head = head->next;
    free(current);// delete current;
    return head;
}

Node* DescCreate(Node* h, Node* p)     该功能占12分
{
    if (!h)                            (在其中空链表处理占1分)
    {
        h = p;
        return h;
    }
    if (p->points > h->points)         (在其中占3分)
    {
        p->next = h;
        h = p;
        return h;
    }//插入头部
    Node* cur = h;                     (在其中查找占5分)
    Node* prev = h;
    while (cur)
    {
        if (p->points > cur->points)
            break;
        prev = cur;
        cur = cur->next;
    }//查找合适的位置，在prev后插入
    p->next = prev->next;              (在其中插入占2分)
    prev->next = p;
    return h;
}
```

（其他做法，比如 第 2 小题 每次插入之后 都排序，如果写对了，也得分）