模拟 C++语言试卷 5

一、单项选择题（共 15 题，每题 2 分，共计 30 分，每题有且仅有一个正确选项）

1. 一幅 256*256 的图像，若灰度级数为 16，则存储它所需的比特数是(　　)。
A. 256kbit　　B. 512kbit　　C. 1M　　D. 2M

2. 下面关于图的论述中哪个是不正确的（　　）?
A. 图由顶点和边的集合组成
B. 有向图中不允许存在没有边或边连接的顶点存在
C. 图中所有顶点的度之和为边的数量的两倍
D. 邻接表是图的一种链式存储结构

3. 用两种颜色去围城一个圈的 6 个棋子，如果通过旋转得到则只算一种，
则一共有多少（　）种染色模式。
A.10　　B.14　　C.15　　D.16

4. 为了保证公司网络的安全运行，预防计算机病毒的破坏，可以在计算机上采取以下哪种方法（　）。
A. 磁盘扫描　　B. 安装浏览器加载项
C. 开启防病毒软件　　D. 修改注册表

5. 十进制小数为 0.96875，对应的二进制数为（　）。
A.0.11111　　B.0.111101
C.0.111111　　D.0.1111111

6. 由 3 个“1”和 5 个“0”组成的 8 位二进制补码，能表示的最小整数
(　　)。
A.-126　　B.-125　　C.-32　　D.-3

7. 安迪-比尔定理指出：硬件提高的性能，很快被（　　）消耗掉了。
A. 操作系统　　B. CPU
C. 计算机病毒　　D. 软件

8. 某二叉树的先序遍历序列和后序遍历序列正好相反，则该二叉树具有的特征是（　　）。
A. 高度等于其结点数
B. 任一结点无左孩子
C. 任一结点无右孩子
D. 空或只有一个结点

9. 一个有序数列，序列中的每一个值都能够被 2 或者 3 或者 5 所整除，这个序列的初始

值从 1 开始，但是 1 并不在这个数列中。求第 1500 个值是多少（　　）。
A.2040　　B.2042　　C.2045　　D.2050

10. 云计算是通过大量在云端的计算资源进行计算，下列（　　）项不是云计算的特点。
A. 超大规模
B. 高可扩展性
C. 无危险性
D. 虚拟化

11. 假定利用数组 a[n] 顺序存储一个栈，用 top 表示栈顶指针，用 top==-1 表示栈空，并已知栈未满，当元素 x 进栈时所执行的操作为（　　）。
A.a[--top]=x
B.a[top--]=x
C.a[++top]=x
D.a[top++]=x

12. 已知声明并初始化二维数组 “int a[3][2]={{1,2},{3,4},{5,6}};”，则 a[1][1] 的值为（　　）。
A.1　　B.2　　C.4　　D.5

13. 十进制数 1000 对应的二进制数和十六进制数为（　　）。
A.1111101010 和 3F8
B.1111101000 和 3E8
C.1111101100 和 3D8
D.1111101110 和 3C8

14. 设有一个递归算法如下。试问计算 f(f(9)) 时需要计算（　　）次 f 函数。

```
int f(int n){
    if(n<=3)
        return 1;
     else
    return f(n-2)+f(n-6)+1;
}
```

A.10　　B.11　　C.12　　D.14

15. 编译器的主要功能是（　　）。
A. 将一种高级语言翻译成另一种高级语言
B. 将源程序翻译成指令
C. 将低级语言翻译成高级语言
D. 将源程序重新组合

二、阅读程序（程序输入不超过数组或字符串定义的范围：判断题正确填 √，错误填×；除特殊说明外，判断题 1.5 分，选择题 3 分，共计 40 分）

1.

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n,k=1;
    cin>>n;
    while (n>k) {
        n=n-k;
        k++;
    }
    if(k%2==0) cout<<n<<"/"<<(k+1-n);
    else cout<<k+1-n<<"/"<<n;
    return 0;
}
```

判断题：

1.程序输出的时候，k+1-n 必定小于 n。（　　）

2.输出一定是一个真分数。（　　）

3. k 的初始值设置为 0 不影响程序结果。（　　）

4. while 判断变成 n>=k，程序结果会受到影响。（　　）

选择题：

5. 若输入 7，程序输出结果为（　　）。

A. 1/4　　B. 2/4　　C. 2/3　　D. 1/3

6. 由程序可以看出，当 n 在什么范围时，分子分母之和为 6（　　）

A. [1,2]　　B. [5,7]　　C. [11,15]　　D. [16,21]

2.

```
#include <bits/stdc++.h>
using namespace std;
int a[1100],p;
int dp[1010];
int dfs(int pos,bool limit) {
    if (pos==-1) return 1;
    if (!limit && dp[pos]!=-1) return dp[pos];
    int u = limit?a[pos]:9;
    int ret = 0;
    for (int i=0; i<=u; ++i) {
        if (i==7) continue;
        ret += dfs(pos-1,limit && i==a[pos]);
```

```
13      }
14      if (!limit) dp[pos] = ret;
15      return ret;
16  }
17  int work(int x) {
18      p = 0;
19      while (x) {
20          a[p++] = x%10;
21          x /= 10;
22      }
23      return dfs(p-1,true);
24  }
25  int main() {
26      memset(dp,-1,sizeof(dp));
27      int a = 0;
28      scanf("%d",&a);
29      printf("%d\n",work(a)-1);
30      return 0;
31  }
```

判断题:

1. 19-22 程序在做数位分离并放到 a 数组中。(　　)

2. dp 数组初始值为 0。(　　)

3. dp 数组作用是记忆化，提高了递归的效率。(　　)

4. 20 行 p++改为++p 不会影响程序结果。(　　)

选择题:

5. 若输入

10

输出结果是(　　)。

A. 9　　B. 10　　C.8　　D. 7

6. 若输入

12345

输出结果是（　　）

A. 7670　　B. 8303　　C.7499　　D. 8534

3.

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   stack<int> n;
```

```
4   char ch;
5   int s,x,y;
6   int main()
7   {
8       while(ch!='@')
9       {
10          ch=getchar();
11          switch(ch)
12          {
13    case '+':x=n.top();n.pop();y=n.top();n.pop();n.push(x+y);break;
14    case '-':x=n.top();n.pop();y=n.top();n.pop();n.push(y-x);break;
15    case '*':x=n.top();n.pop();y=n.top();n.pop();n.push(x*y);break;
16    case '/':x=n.top();n.pop();y=n.top();n.pop();n.push(y/x);break;
17    case '.':n.push(s);s=0;break;
18    default :s=s*10+ch-'0';break;
19          }
20      }
21      printf("%d\n",n.top());
22      return 0;
23  }
```

判断题:

1. 程序使用了队列的数据结构。(　　)

2. 符号越靠后面越早运算。(　　)

选择题:

3. 输入

3.5.2.-*7.+@

输出是(　　)。

A. 15

B. 16

C. 17

D. 18

4. 输入字符'.'的作用是(　　)。

A. 小数点　　B. 区分两个符号

C. 区分两个数字　　D. 区分数字与符号

5. 输入

1.@2.+

输出是(　　)。

A. 1　　B. 2

C. 3　　D.没有输出

6. 输入中出现了除了+-*/.和@之外的符号，输出会（　　）。
A. 程序运行不报错，没有输出
B. 程序运行报错
C. 程序运行不报错，正常输出，会实现额外符号的功能
D. 程序运行不报错，正常输出，但不会实现额外符号的功能

三、完善程序（单选题，每小题 3 分，共计 30 分）

1. （扫雷游戏）
扫雷游戏是一款十分经典的单机小游戏。在 n 行 m 列的雷区中有一些格子含有地雷（称之为地雷格），其他格子不含地雷（称之为非地雷格）。玩家翻开一个非地雷格时，该格将会出现一个数字——提示周围格子中有多少个是地雷格。游戏的目标是在不翻出任何地雷格的条件下，找出所有的非地雷格。
现在给出 n 行 m 列的雷区中的地雷分布，要求计算出每个非地雷格周围的地雷格数。
注：一个格子的周围格子包括其上、下、左、右、左上、右上、左下、右下八个方向上与之直接相邻的格子。

输入：
第一行是用一个空格隔开的两个整数 n 和 m，分别表示雷区的行数和列数。
接下来 n 行，每行 m 个字符，描述了雷区中的地雷分布情况。字符'*'表示相应格子是地雷格，字符'?'表示相应格子是非地雷格。相邻字符之间无分隔符。
输出：
输出文件包含 n 行，每行 m 个字符，描述整个雷区。用'*'表示地雷格，用周围的地雷个数表示非地雷格。相邻字符之间无分隔符。

样例：
输入：
3 3
*??
???
?*?
输出
*10
221
1*1

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  char a[101][101];
4  int b[101][101]={0};
5  int n,m,i,j;
6  int dx[8]={1,0,-1,0,-1,1,1,-1};
7  int dy[8]={0,1,0,-1,-1,-1,1,1};
```

```
8  void dfs(int x,int y)
9  {
10   int nx,ny,k;
11   for (k=0;k<8;k++)
12     {
13           nx=x+dx[k];
14           ny=y+dy[k];
15           if (nx>=1&&nx<=n&&ny>=1&&ny<=m)
16               ---1---;
17     }
18 }
19 int main()
20 {
21   cin>>n>>m;
22   for (i=1;i<=n;i++)
23       for (j=1;j<=m;j++)
24            {
25             cin>>a[i][j];
26             if (---2---)
27                ---3---;
28            }
29   for (i=1;i<=n;i++)
30   {
31       for (j=1;j<=m;j++)
32            if (---4---)
33                cout<<a[i][j];
34           else
35                     cout<<b[i][j];
36            ---5---;
37       }
38   return 0;
39 }
```

1. （1）处应填（　　）。

A. b[nx][ny]--　　B. b[nx][ny]++　　C. b[nx][ny]=0　　D.b[nx][ny]=1

2. （2）处应填（　　）。

A. a[i][j]=='*'　　B. a[i][j]!='*'　　C. i==n　　D. j==m

3. （3）处应填（　　）。

A. dfs(j,j)　　B. dfs(a[i][j])　　C. dfs(i,j)　　D. dfs(i,i)

4. （4）处应填（　　）。

A. b[i][j]!=0　　B. b[i][j]==0　　C. a[i][j]!='*'　　D. a[i][j]=='*'

5. （5）处应填（　　）。
A. break　　B. cout<<endl　　C. cout<<' '　　D. continue

2. （回文数）
若一个数(首位不为 0)从左向右读与从右向左读都一样，我们就将其称为回文数。
例如：给定一个十进制数 56，将 56+65，得到 121 是一个回文数。
又比如对于十进制数 87
1.87+78=165
2.165+561=726
3.726+627=1353
4.1353+3531=4884

在这里一步是指进行了一次 N 进制的加法，上面样例最少使用 4 步的到回文数 4884。

写一个程序，给定一个 N(2<=N<=10 或 N=16)进制数 M(100 位以内)，求最少经过几步可以得到回文数。如果在 30 步以内(包含 30 步)不可能得到回文数，则输出 Impossible!。

输入格式
两行，分别是 N，M

输出格式
如果能在 30 步以内的到回文数输出格式形如 STEP=ans，其中 ans 为最少得到回文数的步数。
否则输出 Impossible!。

样例
输入：
10
87
输出：
STEP=4

```
#include <bits/stdc++.h>
using namespace std;
const int maxN = 105;
char sixt[20] = "0123456789ABCDEF";
int n;
string m;
bool hw(string a) {
    string s = a;
    reverse(s.begin(), s.end()); //reverse 函数的作用是反转字符串 s
    return ---1---;
}
string add(int k, string b) {
```

```
13      string a = b;
14      reverse(a.begin(), a.end());
15      int numa[maxN], numb[maxN], numc[maxN];
16      int len = a.length(), lenc = 1;
17      string ans;
18      for (int i=0; i<len; i++) {
19          if (isdigit(a[i])) numa[len-i] = a[i] - '0'; //isdigit 函数判断是不是数字
20          else numa[len-i] = a[i] - 'A' + 10;
21          if (isdigit(b[i])) numb[len-i] = b[i] - '0';
22          else numb[len-i] = b[i] - 'A' + 10;
23      }
24      int x = 0;
25      while (lenc <= len) {
26          numc[lenc] = ---2---;
27          x = numc[lenc] / k;
28          ---3---;
29          lenc++;
30      }
31      numc[lenc] = x;
32      while (numc[lenc] == 0) lenc--;
33      for (int i=lenc; i>=1; i--) ans +=---4---;
34      return ans;
35  }
36  int main() {
37      cin >> n >> m;
38      for (int i=0; i<=30; i++){
39          if(hw(m)) {
40              printf("STEP=%d", i);
41              return 0;
42          }
43          else m = add(n, m);
44      }
45      printf("---5---");
46      return 0;
47  }
```

1. （1）处应填（　　）。

A. 0　　B. s == a　　C. 1　　D. s!= a

2. （2）处应填（　　）。

A. numa[lenc] + numb[lenc]　　B. numa[lenc] +x

C. numa[lenc] + numb[lenc] + x　　D. numb[lenc]+x

3. （3）处应填（　　）。

A. numc[lenc] %= k　　B. numc[lenc] %= 10　　C. numc[lenc]--　　D. numc[lenc+1]--

4. （4）处应填（　　）。

A. numb[i]　　B. numc[i]

C. numa[i]　　D. sixt[numc[i]]

5. （5）处应填（　　）。

A. Impossible!　　B. Impossible

C. "STEP=%d", N　　D. "STEP=%d", i