模拟 C++语言试卷 6

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. 已知一维数组定义 a：array[1...10000] of int。每个元素占 4 个字节地址。已知 a[1]的开始地址为内存中第 10000 个字节处，请问 a[2020]的开始地址是第几个字节：（　　）。
A. 18072　　B. 18076　　C. 12019　　D. 12020

2. 在 ASCII 码表中，根据码值由小到大的排列顺序是（　　）。
A. 空格字符、数字符、大写英文字母、小写英文字母
B. 数字符、空格字符、大写英文字母、小写英文字母
C. 空格字符、数字符、小写英文字母、大写英文字母
D. 数字符、大写英文字母、小写英文字母、空格字符

3. 由四个完全没有区别的点构成的简单无向连通图的个数是（　　）。
A. 小于 5 个　　B. 5 个　　C. 6 个　　D. 大于 6 个

4. 小明在递归函数内部定义了一个长度为 2020 的 long long 数组，但编译运行时却频频报错，请问最有可能的原因是（　　）。
A. 没有缩进
B. 程序效率太低导致超时
C. 数组命名中包含了大写字母
D. 递归层次过多导致栈空间不足

5. 不同类型的存储器组成了多层次结构的存储器体系，按存取速度从快到慢的排列是（　　）。
A. 快存/辅存/主存
B. 外存/主存/辅存
C. 快存/主存/辅存
D. 主存/辅存/外存

6. 有两个三口之家一起出行去旅游，他们被安排坐在两排座位上，其中前一排有 3 个座位，后面一排有 4 个座位。如果同一个家庭的成员只能被安排在同一排座位并必须相邻而坐，那么共有(　　)种不同的安排方法。
A. 36　　B. 72　　C. 144　　D. 288

7. 有 A，B 和 C 三根柱子，开始时 n 个大小互异的圆盘从小到大叠放在 A 柱上，现要将所有圆盘从 A 移到 C，在移动过程中始终保持小盘在大盘之上，求移动盘子次数的最小值。这类问题叫汉诺塔（Hanoi）问题。求问：对于 n=2020 个圆盘的 Hanoi 塔问题，总的最少移动次数为（　　）。
A. 2^{2019}　　B. 3^{2020}　　C. $2^{2020}-1$　　C. 2^{2020}

8. 中缀表达式(A+B)*(C*(D+E)+F)的后缀表达式是(　　)。
A. ABC*DE+F+*　B. A+BCDE+*F+*　C. AB+C+*DEF+*　D. AB+CDE+*F+*

9. 快速排序（Quicksort），又称分区交换排序（partition-exchange sort），简称快排，是一种排序算法，最早由东尼·霍尔提出。在平均状况下，排序 n 个项目需要 O(nlogn)次比较。在最坏状况下则需要 O(n^2)次比较，但这种状况并不太常见。现在，对给定的整数序列（541,132,984,746,518,181,946,314,205,827）进行从小到大排序时，我们采用快速排序（以中间元素 518 为基准数），请问第一趟扫描的结果是（　）。
A. (181,132,314,205,541,518,946,827,746,984)
B. (541,132,827,746,518,181,946,314,205,984)
C. (205,132,314,181,518,746,946,984,541,827)
D. (541,132,984,746,827,181,946,314,205,518)

10. 10000 以内，与 10000 互质的正整数有（　）个。
A. 2000　B. 4000　C. 6000　D. 8000
答案：B

11. 根节点深度为 0，一颗深度为 h 的满 k（k>1）叉树，也就是说，这棵树除最后一层即叶子层无任何子节点外，每一层上的所有结点都挂满了 k 个子结点。请问这棵树共有（　）个结点。提示：等比数列的求和公式是：(a_0-a_n*q)/(1-q)，其中 q 是等比数列的公比，a_0是首项，an 是数列的末项。
A. (k^{h+1}-1)/(k-1)
B. k^{h-1}
C. k^h
D. k^{h-1}/(k-1)

12. 一个 16 位带符号的整数二进制补码为 1111111111111101，其表示的十进制整数是（　）。
A.-1　B.-2　C.-3　D.-4

13. 计算机术语“中断”是指（　）
A. 操作系统随意停止一个程序运行
B. 当出现需要时，CPU 暂时停止当前程序的执行，转而执行处理新的情况的过程
C. 因停机而停止一个程序的运行
D. 电脑死机

14. 图 G 是一个有序二元组（V，E），其中 V 称为顶点集（Vertices Set），E 称为边集（Edges set）。全部由有方向性的边所构成的图，称为有向图（Directed Graph）。在所有与图中某个点 A 关联的边中，以 A 为起点的边的条数称为出度，而以 A 为终点的边的条数则称为入度。请问：有向图中每个顶点的度等于该顶点的（　）。
A. 入度
B. 出度
C. 入度与出度之和
D. 入度与出度之差

15. 哈夫曼树（Huffman Tree）是在叶子结点和权重确定的情况下，带权路径长度（WPL）最小的二叉树，也被称为最优二叉树。而哈夫曼树所生成的二进制编码，就是哈夫曼编码（Huffman Coding）。这种编码方式由 MIT 的哈夫曼博士发明。它实现了两个目标：一，保证任何字符编码，都不是其他字符编码的前缀；二，通过 WPL 最小保证了信息编码的总长度最小。假设在某次通信中，只用了 5 个字母，a、b、c、d、e，它们出现的频次分别是{2,8,5,4,6}，对其哈夫曼编码的结果是（　　）。

A. {a:010;b:11;c:00;d:011;e:10}

B. {a:00;b:11;c:010;d:011;e:10}

C. {a:010;b:11;c:011;d:00;e:10}

D. {a:11;b:010;c:00;d:011;e:10}

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.

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n,n1,n2;
5      int ans=0;
6      int flag=1;
7      int i=2;
8      cin>>n1>>n2;
9      n=n1+n2;
10     while(i*i<=n){
11         if(n%i==0){
12             flag=0;
13         }
14         i++;
15     }
16     ans=n1*n2*flag;
17     cout<<ans<<endl;
18     return 0;
19 }
```

判断题：

1. 程序第 5 行定义整型变量 ans 时，如果去掉初始化为 0 的赋值操作，对本程序输出结果没有任何影响。（　　）

2. 输入到 n1 和 n2 的值，不允许同时超过 1e9。（　　）

3. 将程序第 7 行定义的整型变量 i 初始化为 1，程序输出结果和修改前一致。（　　）

4. 在程序 12 行和 13 行之间插入一行 break 语句，能够在不影响程序输出结果的前提下，减少程序运行的总耗时。（　　）

选择题：

5. 对第 10 行 while 循环内条件改写成以下哪种形式，程序输出结果一定不变。（　　）

A. i*i*i<=n　　B. i<=n　　C. i<n　　D. i*i>=n

6. 若输入以下哪组 n1 和 n2，程序最终输出结果不是 0。（　　）

A. 4 5　　B. 21 12　　C. 100 189　　D. 45 52

2.

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   string s;
4   long long magic(int l, int r){
5       long long ans = 0;
6       for (int i = l; i <= r; ++i)
7           ans = ans * 4 + s[i]-'a'+1;
8       return ans;
9   }
10  int main(){
11      cin >> s;
12      int len = s.length();
13      int ans = 0;
14      for (int l1 = 0; l1 < len; ++l1)
15          for (int r1 = l1; r1 < len; ++r1){
16              bool bo = true;
17              for (int l2 = 0; l2 < len; ++l2)
18                  for (int r2 = l2; r2 < len; ++r2)
19                      if (magic(l1,r1)==magic(l2,r2)&&(l1!=l2||r1!=r2))
20                          bo = false;
21              if (bo) ans += 1;
22          }
23      cout << ans << endl;
24      return 0;
25  }
```

判断题：

1. 根据程序，如果输入字符串 helloworld，第 12 行整型变量 len 会通过 length 函数的返回值，被初始化为 9。（　　）

2. 假设输入字符串长度为 n，那么本程序因为主函数中包含一个四重 for 循环，因此总时间复杂度可以表示为 $O(n^4)$ 。(　　)

3. 将本程序中 16 行语句放到 13 行和 14 行之间，对输出结果没有影响。(　　)

4. 程序运行过程中，即使 19 行 if 判断条件成立，bo 被置为 false，也会继续进行 17-18 行的双重 for 循环，直到整个字符串的子段被枚举完毕。(　　)

选择题：

5. 输入 abacaba，输出结果是（　　）

A. 7　　B. 7!　　C. 3　　D. 16

5.如果将第 7 行修改为 ans=ans*1+s[i]-'a'+1，输入 abacaba，输出结果与修改前相比(　　)。

A.会变小　　B. 会增大　　C. 程序会报错　　D. 不变

3.

```
1   #include<iostream>
2   #include<cstring>
3   using namespace std;
4   int main() {
5       string a1, b1;
6       int a[109], b[109], c[109];
7       int lena, lenb, lenc;
8       int i, j, x;
9       cin >> a1 >> b1;
10      memset(a, 0, sizeof(a));
11      memset(b, 0, sizeof(b));
12      memset(c, 0, sizeof(c));
13      lena = a1.length();
14      lenb = b1.length();
15      for(i = 0; i <= lena - 1; i++)
16          a[lena - i] = a1[i] - '0';
17      for(i = 0; i <= lenb - 1; i++)
18          b[lenb - i] = b1[i] - '0';
19      for(i = 1; i <= lena; i++) {
20          x = 0;
21          for(j = 1; j <= lenb; j++) {
22              c[i + j - 1] += a[i] * b[j] + x;
23              x = c[i + j - 1] / 10;
24              c[i + j - 1] = c[i + j - 1] % 10;
25          }
26          c[i + lenb] = x;
```

```
27      }
28      lenc = lena + lenb;
29      while(c[lenc] == 0 && lenc > 1)
30          lenc--;
31      for(i = lenc; i >= 1; i--)
32          cout << c[i];
33      cout << endl;
34      return 0;
35  }
```

判断题：

1. 当使用 memset 函数对一个整型数组进行整体地初始化时，通常只能赋值为 0 或-1，因为 memset 函数是按字节（byte）赋值，对每个字节赋值与同样的值。（　　）

2. 可以把 20 行整型变量 x 初始化为 0 的操作，放在 19 行 for 循环的前面，对整个程序输出结果不会有影响。（　　）

选择题：

3. 输入 1234　45678，那么 a[1]和 b[4]的值分别是（　　）。

A. 1 4　　B. 4 4　　C. 4 8　　D. 4 5

4. 输入 1234　45678，程序首次执行到第 26 行时，等价于下面的表达式（　　）。

A. c[7]=1　　B. c[6]=1　　C. c[7]=2　　D. c[6]=2

5. 本程序所实现功能可以概括为（　　）

A. 大整数加法　　B. 大整数乘法　　C. 大整数乘方　　D. 大整数开方

6.（4 分）输入 202020192018　202220212020，将 29 行 while 改成 if，括号内判断条件不变，程序输出结果（　　）。

A. 不变　　B. 变大　　C. 变小　　D. 可能导致溢出

三、完善程序（单选题，每小题 3 分，共计 30 分）

1. （素数环）从 1 到 10 这 10 个数摆成一个环，要求相邻的两个数的和是一个素数。从 1 开始，每个空位有 10 种可能，需要填进去的数合法，即与前面的数不相同，同时与左侧相邻的数之和是一个素数。另外在第 10 个数时还要判断和第 1 个数之和是否是素数。输出每一种可能的排列。

```
1   #include<iostream>
2   #include<cstdio>
3   #include<cstdlib>
4   using namespace std;
5
```

```
bool b[11] = {0};
int total = 0;
int a[11] = {0};

bool prime(int x, int y) {
    int k = 2;
    int i = ____(1)____;
    while(____(2)____)
        k++;
    if(k * k > i)
        return 1;
    else
        return 0;
}

int out() {
    total++;
    cout << " <" << total << "> ";
    for(int j = 1; j <= 10; j++) {
        cout << a[j] << " ";
    }
    cout << endl;
}

void search(int t) {
    int i;
    for(i = 1; i <= 10; i++) {
        if(!b[i] && prime(a[t - 1], i)) {
            a[t] = i;
            b[i] = 1;
            if(t == 10) {
                if(prime(a[____(3)____], a[1]))
                    out();
            }
            else
                ____(4)____;
            ____(5)____;
        }
    }
}

int main() {
    search(1);
    cout << total << endl;
```

```
50      return 0;
51  }
```

1. （1）处应填（　　）。
A. x*y　　B. x+y　　C. x%y　　D. x<<y

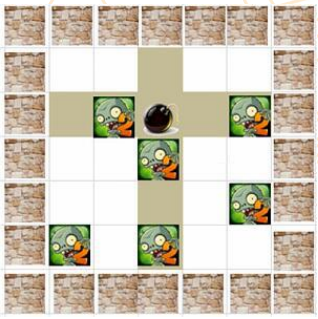
2. （2）处应填（　　）。
A. k*k<=i&&i%k!=0　B. k*k<=i&&i%k==0　C. i%k==0　D. k*k<=i

3. （3）处应填（　　）。
A. 1　　B. 0　　C. 9　　D. 10

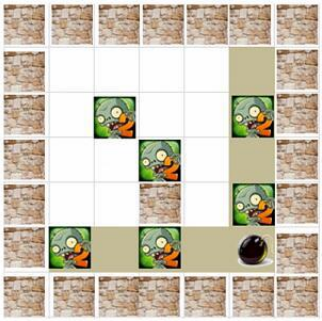
4. （4）处应填（　　）。
A. search(t)　　B. search(t+1)　　C. search(i)　　D. search(i+1)

5. （5）处应填（　　）。
A. b[t]=0　　B. b[i]=0　　C. a[i]=t　　D. a[t]=0

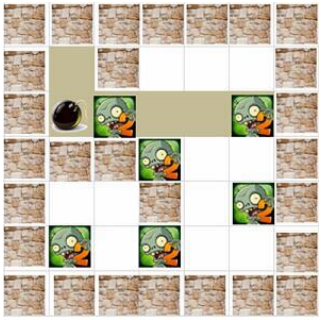
2. （大战僵尸）妈妈得知小新成功地解决了难题，奖励他去看电影《植物大战僵尸》，小新看着看着，就替女主角担心起来了，因为她要对付那么多的僵尸怪物，小新恨不得扔颗炸弹消除可恶的僵尸们，他脑袋里开始构思出这样的场景。



在一个 N 行 M 列单元格构成的地图中，去放置一个炸弹，这种炸弹威力巨大，以放置点为中心进行行列延伸炸到同行同列的僵尸，但不能穿墙。上图中可以把炸弹放置在第 3 行第 4 列，最多可以炸到 4 个僵尸，如果对地图稍加改动（如下图），在第 5 行第 4 列处加入一个墙体，又如何呢？答案其实还是最多炸到 4 个僵尸，只不过此时此刻，最佳炸弹放置点需要发生了变化，应该放到第 6 行第 6 列的位置。



当然炸弹要靠勇敢的小新去放，他只能在地图中朝上下左右四个方向行进（不能斜对角移动），他不能穿墙，也不能穿越僵尸，要保证他安全。如下图，告诉你小新起始位置是第 2 行第 2 列，那么他最佳放置炸弹位置应该是第 3 行第 2 列：最多炸到 2 个僵尸。



现在请聪明的你也一起加入到消除僵尸的队伍来。

输入文件 boom.in

第一行四个用空格隔开的正整数表示 N,M,X,Y，分别表示 N 行 M 列的地图，小星星起始位置第 X 行，第 Y 列。

接下来 N 行 M 列用来描述地图上每个单元格，‘G’表示僵尸，‘#’表示墙体，只有‘.’表示的单元格才是小新能够正常行走，能够放置炸弹的单元格。（数据保证四面都是墙体，也就是第 1 行、第 N 行、第 1 列、第 M 列肯定都是墙体）。

输出文件 boom.out

输出一个整数，最多能炸掉的僵尸数量。

输入样例

13 13 4 2
#############
###..GG#GGG.#
###.#G#G#G#G#
#.....#..G#
#G#.###.#G#G#
#GG.GGG.#.GG#
#G#.#G#.#.#.#
#G···G····.#
#G#.#G###.#G#
#···G#GGG.GG#
#G#.#G#G#.#G#
#GG.GGG#G.GG#
#############

输出样例

10

数据规模

30%的数据，保证 N,M<14，并且小新一定能够抵达最佳炸弹放置点
40%的数据，保证 N,M<14
70%的数据，保证 N,M<101
100%的数据，保证 N,M<2001
100%的数据，保证 X<N

100%的数据，保证 Y<M

小新苦心冥想，写出了如下代码，但有些地方还不太确定。请帮帮他，把代码补全。

```
#include<bits/stdc++.h>
using namespace std;
const int SIZE=10009;
struct node{
    int x,y;
};
char a[SIZE][SIZE];
bool vst[SIZE][SIZE];
int ans=0;
int dx[4]={-1,1,0,0};
int dy[4]={0,0,-1,1};
node _next;
queue<node> q;
//上：该点之上能炸到几个僵尸
int _up(int x,int y){
    if(a[x][y]=='#') //障碍则停
        return 0;
    if(a[x][y]=='G') //炸到僵尸
        return ____(1)____; //
    return _up(x-1,y);
}
//下：该点之下能炸到几个僵尸
int _down(int x,int y){
    if(a[x][y]=='#')
        return 0;
    if(a[x][y]=='G')
        return 1+_down(x+1,y);
    return _down(x+1,y);
}
//左
int _left(int x,int y){
    if(a[x][y]=='#')
        return 0;
    if(a[x][y]=='G')
        return 1+_left(x,y-1);
    return _left(x,y-1);
}
//右
int _right(int x,int y){
    if(a[x][y]=='#')
        return 0;
```

```
42      if(a[x][y]=='G')
43          return 1+_right(x,y+1);
44      return _right(x,y+1);
45  }
46  //统计炸到僵尸数目
47  int count(int x,int y){
48      return ____(2)____;
49  }
50  //广搜求最大值
51  void bfs(){
52      while(!q.empty()){ //队列 q 为空则停止
53          node now=q.front(); //队首元素
54          ans=max(ans, count(now.x, now.y)); //打擂台
55          for(____(3)____){ //int i=0;i<4;i++
56              //四个方向扩展
57              int r=now.x+dx[i];
58              int c=now.y+dy[i];
59              if(____(4)____){
60                  _next.x=r;
61                  _next.y=c;
62                  vst[r][c]=true; //标记已访问
63                  q.push(_next);
64              }
65          }
66          q.pop();
67      }
68  }
69
70  int main(){
71      freopen("boom.in","r",stdin);
72      freopen("boom.out","w",stdout);
73      int n,m,x,y;
74      cin>>n>>m>>x>>y;
75      //读入地图
76      for(int i=1;i<=n;i++)
77          for(int j=1;j<=m;j++)
78              cin>>a[i][j];
79      memset(vst,0,sizeof(vst)); //初始化标记用布尔数组
80      _next.x=x;
81      _next.y=y;
82      vst[x][y]=true;
83      q.push(_next);
84      bfs();
85      cout<<____(5)____<<endl;
```

```
86      return 0;
87  }
```

1. （1）处应填（　　）。

A. up(x-1,y)　　B. up(x+1,y)　　C. 1+_up(x-1,y)　　D. 1+_up(x,y+1)

2. （2）处应填（　　）。

A. _up(x,y)+_down(x,y)+_left(x,y)+_right(x,y)

B. _up(x+1,y)+_down(x-1,y)+_left(x,y+1)+_right(x,y-1)

C. _up(x,y-1)+_down(x,y+1)+_left(x-1,y)+_right(x+1,y)

D. _up(x-1,y)+_down(x+1,y)+_left(x,y-1)+_right(x,y+1)

3. （3）处应填（　　）。

A. int i=0;i<4;i++

B. int i=1;i<=4;i++

C. int i=x;i<=y;i++

D. !q.empty()

4. （4）处应填（　　）。

A. !vst[r][c]&&a[r][c]=='G'

B. vst[r][c]&&a[r][c]=='#'

C. !vst[r][c]&&a[r][c]=='.'

D. vst[r][c]&&(a[r][c]=='#'|| a[r][c]=='G')

5. （5）处应填（　　）。

A. vst[x][y]

B. a[x][y]

C. q.front()

D. ans