CCF 编程能力等级认证
Grade Examination of Software Programming

Python 七级（样题）

1 单选题（每题 2 分，共 30 分）

题号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
答案	A	B	C	D	B	D	B	A	C	C	D	C	D	B	B

第 1 题 如果下面代码输入整数为10，输出是1.0，则横线处填写？（ ）

```
import math
x = int(input())
print(__________)
```

- **A.** math.log10(x)
- **B.** math.log(x)
- **C.** math.exp(x)
- **D.** math.pow(x,10)

第 2 题 下面定义的函数用来求斐波那契数列的F(n)，描述正确的是（ ）。

```
def fab(n):
    f = [None] * (n+1)

    f[0], f[1] = 0, 1
    for i in range(2,n+1):
        f[i] = f[i - 1] +f[i - 2]
    return f[n]
```

- **A.** f[0]和f[1]是递归终止条件
- **B.** 数组f保存算法执行过程的状态
- **C.** 使用倍增法来求解
- **D.** 算法不能正常结束

第 3 题 下列关于Python语言中函数的叙述，正确的是（ ）。

- **A.** 新定义函数不能与已有函数名称相同
- **B.** 函数调用前必须定义
- **C.** 在新定义函数中，不能嵌套定义新的函数
- **D.** 函数的返回值不可以是函数的名称

第 4 题 4个结点的简单有向图，最多可以有多少条边（ ）。

- **A.** 4
- **B.** 6

☐ **C.** 8

☐ **D.** 12

第 5 题 哈希表上可以执行的操作不包括（ ）

☐ **A.** 插入

☐ **B.** 排序

☐ **C.** 查找

☐ **D.** 删除

第 6 题 将关键码集合{100，300，500，700，800，900}逐一保存在一个长度为100的哈希表中，选取哈希函数为Hash(key)=key/100，则800保存在表中的位置应该是（ ）。

☐ **A.** 5

☐ **B.** 6

☐ **C.** 7

☐ **D.** 8

第 7 题 在Python中，pi=3.14且变量x已赋值，x代表等边三角形边长，则该三角形的面积是（ ）。

☐ **A.** x*x*sin(pi/3)

☐ **B.** x*x*sin(pi/3)/2

☐ **C.** x*x*cos(pi/3)

☐ **D.** x*x*cos(pi/3)/2

第 8 题 动态规划将一个问题分解为一系列子问题后来求解。下面关于子问题的描述正确的是（ ）。

☐ **A.** 具有重叠子问题的性质

☐ **B.** 和分治法的子问题类似

☐ **C.** 不具有最优子结构的性质

☐ **D.** 问题的最优解可以由部分子问题的非最优解推导出来

第 9 题 阅读以下代码，visited起到的作用是（ ）。

```
visited, k = [None]*100, 0

def dfs(graph, start, vexnum):
    k += 1
    visited[start] = k
    print(start)

    for i in range(vexnum):
        if (start != i) and (not visited[i]) :
            dfs(graph, i, vexnum)
```

☐ **A.** 实现遍历过程。

☐ **B.** 以广度优先的方式记录图中的顶点。

☐ **C.** 存储深搜时节点的访问顺序。

☐ **D.** 能够记录最短路径。

第 10 题 下面函数尝试使用动态规划方法求出如下递推公式的函数，则横线处填写下列哪段代码可以完成预期功能？（ ）

$$C(n,m)=\begin{cases}1 & \text{当 } n=0 \text{ 或 } m=0\\ C(n-1,m-1)+C(n-1,m) & \text{当 } n>0 \text{ 且 } m>0\end{cases}$$

```
MAXN, MAXM = map(int,input().split(","))

rec_C = [[0]*MAXM for i in range(MAXN)]

def C(n, m):
    for i in range(n + 1):
        rec_C[i][0] = 1
    for j in range(m + 1):
        rec_C[0][j] = 1

    _____________________: #此处填入代码

        _____________________: #此处填入代码

            rec_C[i][j] = rec_C[i - 1][j - 1] + rec_C[i-1][j]

    return rec_C[n][m]
```

☐ **A.** for i in range(n, 0, -1)和for j in range(1, m + 1)

☐ **B.** for i in rnage(1, n + 1)和for j in range(m, 0, -1)

☐ **C.** for j in range(1, m + 1)和for i in range(1,n + 1)

☐ **D.** for j in range(1, m + 1)和for i in range(n, 0, -1)

第 11 题 深度为4的完全二叉树，结点总数最少有多少个？（ ）

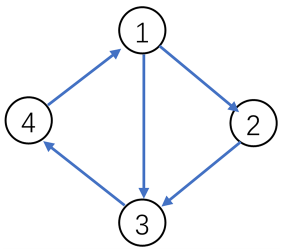
☐ **A.** 5

☐ **B.** 6

☐ **C.** 7

☐ **D.** 8

第 12 题 下面有向图中的数字表示顶点序号，则从1号顶点出发的BFS遍历的输出顶点序列可能是（ ）。



☐ **A.** 1 4 3 2

☐ **B.** 1 4 2 3

☐ **C.** 1 3 2 4

☐ **D.** 1 2 4 3

第 13 题 一个简单有向图有20个结点，假设图中已经存在300条边，请问增加多少条边可以成为完全图。（ ）

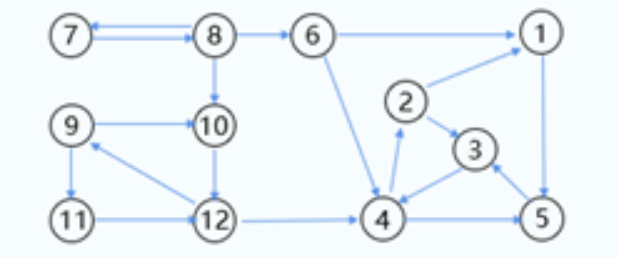
☐ **A.** 77

☐ **B.** 78

☐ **C.** 79

☐ **D.** 80

第 14 题 在下面的有向图中，强连通分量有多少个？（ ）



☐ **A.** 3

☐ **B.** 4

☐ **C.** 5

☐ **D.** 6

第 15 题 下面有关格雷码的说法，错误的是（ ）。

☐ **A.** 在格雷码中，任意两个相邻的代码只有一位二进制数不同

☐ **B.** 格雷码是一种唯一性编码

☐ **C.** 在格雷码中，最大数和最小数只有一位二进制数不同

☐ **D.** 格雷码是一种可靠性编码

2　判断题（每题 2 分，共 20 分）

题号	1	2	3	4	5	6	7	8	9	10
答案	×	×	√	×	×	√	×	√	×	×

第 1 题 假设Python代码已执行import math，以及x=exp(-1)，则x < 0为真。( )

第 2 题 假设x和y中都是浮点型正数，如果说x比y大一个数量级，math.log(x/y)等于10。( )

第 3 题 如果浮点型变量x代表锐角对应的弧度角，则可以编程来确定math.sin(x)>math.cos(x)的近似区间。( )

第 4 题 Python表达式pow(1,2)的值为浮点类型。( )

第 5 题 如果哈希表足够大，哈希函数确定后，不会产生冲突。( )

第 6 题 动态规划最终要推导出状态转移方程才能求解。( )

第 7 题 简单有向图的深搜结果和广搜结果一样。( )

第 8 题 判断图是否连通可以用深搜实现。( )

第 9 题 在Python中，可以使用二分法查找链表中的元素。( )

第 10 题 在C/C++语言中有些算法或数据结构使用指针实现，一个典型的例子就是链表。由于Python没有指针，因此，这类数据结构不能用Python代码实现。( )

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

- **试题名称**：迷宫统计
- **时间限制**：1.0 s
- **内存限制**：128.0 MB

3.1.1 问题描述

在神秘的幻想大陆中，存在着 n 个古老而神奇的迷宫，迷宫编号从 1 到 n。有的迷宫之间可以直接往返，有的可以走到别的迷宫，但是不能走回来。玩家小杨想挑战一下不同的迷宫，他决定从 m 号迷宫出发。现在，他需要你帮助他统计：有多少迷宫可以直接到达 m 号迷宫，m 号迷宫可以直接到达其他的迷宫有多少，并求出他们的和。

需要注意的是，对于 $i\,(1 \leq i \leq n)$ 号迷宫，它总可以直接到达自身。

3.1.2 输入描述

第一行两个整数 n 和 m，分别表示结点迷宫总数 n，指定出发迷宫的编号 m。
下面 n 行，每行 n 个整数，表示迷宫之间的关系。对于第 i 行第 j 列的整数，1 表示能从 i 号迷宫直接到达 j 号迷宫，0 表示不能直接到达。

3.1.3 输出描述

一行输出空格分隔的三个整数，分别表示迷宫 m 可以直接到达其他的迷宫有多少个，有多少迷宫可以直接到达 m 号迷宫，这些迷宫的总和。

3.1.4 样例输入1

```
6 4
1 1 0 1 0 0
0 1 1 0 0 0
1 0 1 0 0 1
0 0 1 1 0 1
0 0 0 1 1 0
1 0 0 0 1 1
```

3.1.5 样例输出1

```
3 3 6
```

3.1.6 样例解释

4 号迷宫能直接到达的迷宫有 $3, 4, 6$ 号迷宫，共 3 个。

能直接到达 4 号迷宫的迷宫有 $1, 4, 5$ 号迷宫，共 3 个。

总和为 6。

3.1.7 子任务

子任务编号	分值	n
1	30	≤ 10
2	30	≤ 100
3	40	≤ 1000

对于全部数据，保证有 $4 \leq n \leq 1000$，$1 \leq m \leq n$。

3.1.8 参考程序

```cpp
#include<cstdio>
#define M  101

int g[M][M];
int main(){
    int n,m;
   scanf("%d %d",&n,&m);
   for(int i=1;i<=n;i++){
   for(int j=1;j<=n;j++){
           scanf("%d",&g[i][j]);
    }
   }
   int ans1 = 0,ans2 = 0;
   for(int i=1;i<=n;i++){
       ans1+=g[i][m];
       ans2+=g[m][i];
   }
   printf("%d %d %d",ans2,ans1,ans1+ans2);
   return 0;
}
```

3.2 编程题 2

- **试题名称**：最长不下降子序列
- **时间限制**：1.0 s
- **内存限制**：128.0 MB

3.2.1 问题描述

小杨有一个包含 n 个节点 m 条边的有向无环图，其中节点的编号为 1 到 n。

对于编号为 i 的节点，其权值为 A_i。对于图中的一条路径，根据路径上的经过节点的先后顺序可以得到一个节点权值的序列，小杨想知道图中所有可能序列中最长不下降子序列的最大长度。

注：给定一个序列 S，其最长不下降子序列 S' 是原序列中的如下子序列：整个子序列 S' 单调不降，并且是序列中最长的单调不降子序列。例如，给定序列 $S = [11, 12, 13, 9, 8, 17, 19]$，其最长不下降子序列为 $S' = [11, 12, 13, 17, 19]$，长度为 5。

3.2.2 输入描述

第一行包含两个正整数 n, m，表示节点数和边数。第二行包含 n 个正整数 $A_1, A_2, \ldots, A_n$，表示节点 1 到 n 的点权。之后 m 行每行包含两个正整数 u_i, v_i，表示第 i 条边连接节点 u_i 和 v_i，方向为从 u_i 到 v_i。

3.2.3 输出描述

输出一个正整数，表示该图中所有可能序列中最长不下降子序列的最大长度。

3.2.4 样例输入1

```
5 4
2 10 6 3 1
5 2
2 3
3 1
1 4
```

3.2.5 样例输出1

```
3
```

3.2.6 样例输入2

```
6 11
1 1 2 1 1 2
3 2
3 1
5 3
4 2
2 6
3 6
1 6
4 6
1 2
5 1
5 4
```

3.2.7 样例输出2

```
4
```

3.2.8 样例输入3

```
6 11
5 9 10 5 1 6
5 4
5 2
4 2
3 1
5 3
6 1
4 1
4 3
5 1
2 3
2 1
```

3.2.9 样例输出3

```
4
```

3.2.10 子任务

子任务编号	分值	n	$\max A_i$
1	30	$\leq 10^3$ 有向无环图为一条链	≤ 10
2	30	$\leq 10^5$	≤ 2
3	40	$\leq 10^5$	≤ 10

对于全部数据，保证有 $1 \leq n \leq 10^5$，$1 \leq m \leq 10^5$，$1 \leq A_i \leq 10$。

3.2.11 参考程序

```cpp
#include "bits/stdc++.h"
using namespace std;
const int N = 100010;
vector<int> g[N];
int n,m;
int a[N];
int d[N],dp[N][15];

int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        cin>>a[i];
    }
    for(int i=1;i<=m;i++){
        int u,v;
        cin>>u>>v;
        g[u].push_back(v);
        d[v]++;
    }
    queue<int> q;
    for(int i=1;i<=n;i++){
        if(!d[i]){
            q.push(i);
            dp[i][a[i]]=1;
```

```
        }
    }
    while(!q.empty()){
        int now = q.front();
        q.pop();
        for(auto i:g[now]){
            for(int j=1;j<=10;j++){
                dp[i][j]=max(dp[i][j],dp[now][j]);
            }
            d[i]--;
            if(!d[i]){
                for(int k=a[i];k>=1;k--){
                    dp[i][a[i]]=max(dp[i][a[i]],dp[i][k]+1);
                }
                q.push(i);
            }
        }
    }
    int mx=1;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=10;j++){
            mx=max(mx,dp[i][j]);
        }
    }
    cout<<mx<<"\n";
}
```