2025年广西中小学生程序设计挑战赛初赛
进阶组试题

比赛时间：2025年4月26日9:00～11:00

考生注意事项：

●试题纸共有12页，答题卡共有1页，满分100分。请在答题卡上作答，写在试题纸上的一律无效。

●不得使用任何电子设备（如计算器、手机、电子词典等）或查阅任何书籍资料。

一、单项选择题（共30小题，每题1.5分，共计45分；每题有且仅有一个正确选项）

1.现代计算机产生于抽象的（　　）机。

A.图灵　　B.冯诺依曼　　C.晶体管　　D.电子管

2.下列关于 C++ 语言特性的描述，正确的是（　　）。

A.C++是一种面向对象的编程语言

B.C++不支持指针操作

C.C++不可以直接操作硬件

D.C++没有模板机制

3.阅读下述代码，说法错误的是（　　）。

```
#define MAX_SIZE 100
int main() {
    const int minSize = 10;
    int values[MAX_SIZE];
    for (int i = minSize; i < MAX_SIZE; i++) {
        values[i] = i;
    }
}
```

A.MAX_SIZE是通过宏定义的常量，它在预处理阶段会被替换

B.minSize是使用const定义的常量，其值不可改变

C.values是数组变量，它的大小由MAX_SIZE决定

D.for循环中的i是常量，因为它在循环中有固定的取值范围

4.若某字符的ASCII码用八进制表示是101，它对应的二进制表示是什么；这个字符是（　　）。

A.1000001；A　　B.1000010；B　　C.1000100；D　　D.1001000；H

5.下列关于算法概念的叙述，错误的是（　　）。

A.算法是解决某一特定类型问题的有限运算序列

B. 算法是指令的有限序列，其中每一条指令表示一个或多个操作

C. 算法可以用自然语言、流程图、伪代码等多种形式来描述

D. 一个有效的算法至少要有一个输入

6. 若要存储一个 0 到 255 之间的整数，满足存储需求且不浪费存储空间的数据类型是（　　）。

A. signed char　　B. unsigned char　　C. Short　　D. int

7. 计算机中用于存储正在运行的程序和数据的部件是（　　）。

A. 硬盘　　B. 内存　　C. 光盘　　D. 软盘

8. 下面四个无符号整数中，（　　）超过了一个字节的表示范围。

A. $(231)_{10}$　　B. $(257)_{8}$　　C. $(102)_{16}$　　D. $(111)_{2}$

9. 下列程序执行后输出的结果是（　　）。

```
char str1[10] = "ABCDE", str2[10] = "xyz";
printf("%d", strlen(strcpy(str1 , str2)));
```

A. 9　　B. 8　　C. 5　　D. 3

10. 对于一个边权都为 1 的无向图，下列哪种算法不能直接或间接用于检测连通性问题（　　）

A. bfs 算法进行图的遍历

B. kruscal 算法求最小生成树

C. dijkstra 算法求最短路径

D. Topological Sorting 算法求拓扑排序

11. 设 int m = 8, n = 2; 则表达式 m - n++的值是（　　）。

A. 5　　B. 6　　C. 7　　D. 8

12. 在编写程序计算从 n 个不同元素中取出 m 个元素的排列数时，以下哪个公式是正确的程序实现依据（　　）。

A. $A_n^m = \frac{n!}{m!}$

B. $A_n^m = \frac{n!}{(n-m)!}$

C. $A_n^m = \frac{(n-m)!}{n!}$

D. $A_n^m = \frac{m!}{n!}$

13. 若要计算一个数的平方根，在 C++中应使用哪个数学库函数（　　）。

A. pow()　　B. sqrt()　　C. log()　　D. sin()

14. 若一个栈的输入序列为 1, 2, 3, 4，不可能得到的输出序列是（　　）。

A. 4, 3, 2, 1　　B. 3, 4, 2, 1　　C. 2, 4, 1, 3　　D. 2, 3, 4, 1

15. 阅读下列代码，该程序中，输出语句一共执行了（　　）次。

```
for(int i = 0, j = 1; i < 100; ++i, ++j, i += j) printf ("%d\n", i);
```

A. 10　　B. 11　　C. 12　　D. 13

16. 已知 int p = 7, q = 3; 表达式 p > q ? p : q 的值是（　　）。

A. 3　　B. 7　　C. 10　　D. 4

17. 在C++中，下面程序段执行后，程序的输出是（　　）。

```
#include<iostream>
#include<cstdio>
int work(int x, int y)
{
  return (y == 0)?x:work(y, x % y);
}
int main()
{
  printf("%d", work(30,42));
  return 0;
}
```

A. 5　　B. 6　　C. 7　　D. 3

18. 若 s="2025gxcxsjtzsjjz"，其不重复子串个数是（　　）。

A. 129　　B. 240　　C. 136　　D. 120

19. 在C++中，下面程序段执行后，变量 count 的值是（　　）。

```
int count = 0;
for (int i = 1; i <= 6; i++) {
    for (int j = 1; j <= 6; j++) {
        int num = i * j * 2;
        if (num % 6) {
            count++;
        }
    }
}
```

A. 4　　B. 8　　C. 12　　D. 16

20. 数组 A 和数组 B 的数据存放情况如下表，则 A[B[8]*A[B[3]]]的值是（　　）。

A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]
1	7	7	1	3	4	8	2	8	10
B[1]	B[2]	B[3]	B[4]	B[5]	B[6]	B[7]	B[8]	B[9]	B[10]
9	2	2	1	9	7	4	1	6	0

A. 8　　B. 6　　C. 4　　D. 2

21. 若 p 是大于 3 的质数，则 $p^2 - 1$ 一定能被（　　）整除。

A. 10　　B. 7　　C. 16　　D. 24

22. 杀毒软件属于（　　）。

A. 系统软件　　B. 应用软件　　C. 开发软件　　D. 管理软件

23. 以下表达式的值为真的是（　　）。

A.!(5>3)　　B.(3==5) && (2<4)　　C.(3<5) || (2>4)　　D.!(3<=5)

24.以下关于流程图的优点，表述错误的是（　　）。

A.有助于发现程序设计中的逻辑错误

B.可以直接运行实现程序功能

C.便于不同人员之间的交流和沟通

D.能清晰地展示程序的整体架构

25.在有向图中，所有顶点的入度之和与出度之和的关系是（　　）。

A.入度之和大于出度之和　　B.入度之和小于出度之和

C.入度之和等于出度之和　　D.无法确定

26.以下关于文本文件和二进制文件读写的区别，说法错误的是（　　）。

A.文本文件读写时，系统会对数据进行格式转换，二进制文件不会

B.二进制文件读写速度通常比文本文件快

C.文本文件只能存储字符数据，二进制文件只能存储数值数据

D.读取文本文件时，getline()函数可用于读取一行数据，而二进制文件读取通常使用read()函数

27.以下关于多层条件语句的说法，错误的是（　　）。

A.多层条件语句可以嵌套使用，以实现复杂的逻辑判断

B.嵌套的条件语句层次越多，代码的可读性和可维护性越好

C.每个if语句都可以有对应的else语句

D.多层条件语句的执行顺序是从上到下，依次判断每个条件

28.下面（　　）算法的时间复杂度在平均情况是$O(n\log n)$。

A.冒泡排序　　B.插入排序　　C.快速排序　　D.选择排序

29.以下递归函数用于计算斐波那契数列的第n项，代码中空白处应填入（　　）。

```
int fibonacci(int n) {
    if (n == 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    } else {
        return _____;
    }
}
```

A.fibonacci(n)

B.fibonacci(n - 1)

C.fibonacci(n - 1) + fibonacci(n - 2)

D.fibonacci(n - 2)

30.任何一个无向连通图的最小生成树（　　）。

A.有且仅有一棵　　B.有一棵或多棵　　C.一定有多棵　　D.可能不存在

二、判断题（共 10 题，每题 1.5 分，共计 15 分；正确选 A，错误选 B）

31. IPv4 地址由 32 位二进制数组成，通常用点分十进制表示法来书写。（　　）

32. 若 x = 5, y = 3, z = 7，表达式(x > y && y < z) || (x < z && !(y > z))的值为真。（　　）

33. 局部变量的作用域是整个程序，在程序的任何地方都可以访问。（　　）

34. 在 C++的 STL 中，min 函数只能用于比较两个整数的大小。（　　）

35. 在 C++中，if 语句后面的条件表达式必须用括号括起来。（　　）

36. 显式类型转换也被称为强制类型转换，它可以将任何类型转换为其他类型。（　　）

37. 1KB（千字节）等于 1000 字节。（　　）

38. 简单无向图的顶点度数之和等于边数的两倍。（　　）

39. 若完成一件事需要分成 n 个步骤，做第 1 步有 m_1 种不同的方法，做第 2 步有 m_2 种不同的方法，……，做第 n 步有 m_n 种不同的方法，那么完成这件事共有 $N=m_1\times m_2\times\cdots\cdots\times m_n$ 种不同的方法。（　　）

40. 汉诺塔问题只能用递归算法解决，不能用其他方法实现。（　　）

三、阅读程序（两小题，共 10 空，每空 2 分，共计 20 分；每空有且仅有一个正确选项）

（1）

```
01  #include <iostream>
02  #include<cstdio>
03  using namespace std;
04  int main() {
05      int k, n;
06      scanf("%d %d", &k, &n);
07      int nums[100];
08      for(int i = 0; i < n; ++i)
09      {
10          scanf("%d", &nums[i]);
11      }
12      int left = 0;
13      int sum = 0;
14      int minLength = n + 1;
15      for(int right = 0; right < n; ++right)
16      {
17          sum += nums[right];
18          while(left <= right && sum >= k)
19          {
20              if(minLength > right -  left + 1)
```

```
21              {
22                  minLength = right -  left + 1;
23              }
24          sum -= nums[left];
25          left += 1;
26        }
27    }
28    if(minLength == n + 1)
29    {
30       printf("%d", 0);
31     }
32     else
33     {
34        printf("%d", minLength);
35     }
36     return 0;
37 }
```

41. 将第 28 行改为 if(minLength > n)，程序输出结果（　　）。

A. 会改变　　B. 不会改变

42. 上述代码所用的算法思想为（　　）。

A. 倍增法　　B. 贪心法　　C. 尺取法　　D. 动态规划

43. 若输入 10 6 3 2 6 2 1 3，该程序的输出为（　　）。

A. 5　　B. 4　　C. 3　　D. 2

44. 若输入 20 25 7 2 1 2 3 2 4 5 1 2 3 4 2 1 4 5 6 2 1 2 3 2 1 5 2，该程序的输出为（　　）。

A. 5　　B. 6　　C. 7　　D. 8

45. 该程序的时间复杂度为（　　）。

A. O(n)　　B. O(nlogn)　　C. O(n^2)　　D. O(n^3)

(2)

```
01  #include <bits/stdc++.h>
02  int interpolationSearch(const std::vector<int> &arr, int target) {
03      int low = 0;
04      int high = arr.size() - 1;
05      while (low <= high && target >= arr[low] && target <= arr[high]) {
06          if (low == high) {
07              if (arr[low] == target)
```

```
            return low;
        return -1;
    }
    int pos = low + (((target - arr[low]) * (high - low)) / (arr[high] - arr[low]));
    if (arr[pos] == target)
        return pos;
    if (arr[pos] < target)
        low = pos + 1;
    else
        high = pos - 1;
  }
  return -1;
}
int main() {
    std::vector<int> arr = {10, 12, 13, 16, 18, 19, 20, 21, 22, 23, 24, 33, 35, 42, 47};
    int target = 18;
    int result = interpolationSearch(arr, target);
    if (result != -1)
        std::cout << "Element found at index " << result << std::endl;
    else
        std::cout << "Element not found in the array." << std::endl;
    return 0;
}
```

46. 插值查找算法适用于以下哪种情况（　　）。

A. 无序数组

B. 有序数组且元素分布均匀

C. 有序数组但元素分布不均匀

D. 任何数组

47. 在 interpolationSearch 函数中，pos 变量的作用是什么（　　）。

A. 记录查找范围的起始位置

B. 记录查找范围的结束位置

C. 计算并记录本次查找的中间位置

D. 记录目标元素的位置

48. 若 arr 数组为{2,4,6,8,10,12,14}，目标值 target 为 8，在第一次计算 pos 时(即初始 low= 0,high=6)，pos 的值是（　　）。

A. 2　　B. 3　　C. 4　　D. 5

49. 插值查找算法在最坏情况下的时间复杂度是（　　）。

A. O(1)　　B. O(log log n)　　C. O(log n)　　D. O(n)

50. 在 interpolationSearch 函数中，当 arr[pos]<target 时，执行 low =pos+1，这一步的目的是（　　）。

A. 扩大查找范围到整个数组

B. 缩小查找范围到 pos 之后的部分

C. 缩小查找范围到 pos 之前的部分

D. 重新开始查找

四、完善程序（两小题，共 10 空，每空 2 分，共计 20 分；每题有且仅有一个正确选项）

（1）

给定一个无向图，判断是否连通，是否有回路。

输入：第一行两个整数 n,m，分别表示顶点数和边数。顶点编号从 0 到 n-1 (1<=n<=110,1<=m<=10000)

接下来 m 行，每行两个整数 u 和 v，表示顶点 u 和顶点 v 之间有边。

输出：

-如果图是连通的，则在第一行输出“connected:yes”，否则输出“connected:no”

-如果图中有回路，则在第二行输出“loop:yes”，否则输出“loop:no”。

```
01  #include <iostream>
02  #include <cstdio>
03  #define maxN 120
04  using namespace std;
05  int G[maxN][maxN], n,m, visited[maxN];
06  int total = 0;
07  void dfsConnection(int v){
08    visited[v] = 1;
09    total += 1;
10    int u;
11    for(u = 0;u < n; ++u)
12    {
13       if (G[v][u] && !visited[u])
14          dfsConnection(u);
15     }
16  }
17  int dfsLoop(int v,int x)//深度优先遍历图寻找环，x 是深度优先搜索树上 v 的父结点
18  {
19    visited[v] = 1;
20    int u;
```

```
21    for(u = 0;u < n; ++u)
22    {
23        if ( G[v][u] )
24        {
25          if (!visited[u]) {
26            if (   ①   ) return 1;
27              }
28            else
29               if (   ②   ) return 1;
30         }
31    }
32    return 0;
33  }
34  int main()
35  {
36     scanf("%d%d",&n,&m);
37     int i;
38     memset(G,0,sizeof(G));
39     memset(visited,0,sizeof(visited));
40     for(i = 0; i< m; ++i)
41     {
42        int a,b;
43        scanf("%d%d",&a,&b);
44        G[a][b] = G[b][a] = 1;
45     }
46    total = 0;
47    dfsConnection(0);
48    if(   ③   )
49       printf("connected:yes\n");
50    else
51       printf("connected:no\n");
52    memset(visited,0,sizeof(visited));
53    int loopFound = 0;
54    for(i=0;i<n;++i)
55    if (   ④   )
56       if (   ⑤   )
57       {
```

```
58          loopFound = 1;
59          break;
60        }
61  if (loopFound)
62      printf("loop:yes\n");
63  else
64  printf("loop:no\n");
65  return 0;
66  }
```

51. ① 处的代码为（　）

A. dfsLoop(u, v)　　B. dfsLoop(u, -1)　　C. !dfsLoop(u, v)　　D. !dfsLoop(u, -1)

52. ② 处的代码为（　）

A. !visited[x]　　B. visited[x]　　C. u == x　　D. u!= x

53. ③ 处的代码为（　）

A. total <= n　　B. total <= n - 1　　C. total == n　　D. total == n - 1

54. ④ 处的代码为（　）

A. visited[i]　　B. !visited[i]　　C. visited[i+1]　　D. !visited[i+1]

55. ⑤ 处的代码为（　）

A. dfsLoop(i, 0)　　B. dfsLoop(i, -1)　　C. dfsLoop(0, -1)　　D. dfsLoop(0, 0)

（2）

明天就是劳动节了，电脑组的小杨在忙碌的课业之余，满心想着要给辛苦工作的父母送份贴心的礼物，以表感恩之情。听说在某个神奇商店上有卖云朵的，这可把小杨吸引住了，决定前往一探究竟，看看这种神奇的商品。这个店里有 n 朵云，云朵已经被老板编号为 1,2,3,...,n，并且每朵云都有一个价值。然而，商店的老板是个很古怪的人，他规定一些云朵要搭配起来才出售，也就是说买一朵云的话，与之有搭配关系的云都得一并购买。这种搭配是具有传递性的，即若 A 要搭配 B 出售，那么 B 也要搭配 A 出售。小杨觉得这礼物实在新奇，可自己手头的钱有限，因此一心想用现有的钱买到价值总和尽量高的云朵组合，给父母送上一份独特又珍贵的礼物。

输入：

第一行输入三个整数，n,m,w，表示有 n 朵云，m 个搭配和你现有的钱的数目。

第二行至 n+1 行，每行有两个整数，c_i,d_i，表示第 i 朵云的价钱和价值。

第 n+2 至 n+1+m 行，每行有两个整数 u_i,v_i。表示买第 u_i 朵云就必须买第 v_i 朵云，同理，如果买第 v_i 朵就必须买第 u_i 朵。

输出：

一行，表示可以获得的最大价值。

程序如下：

```
01 #include<bits/stdc++.h>
02 #define N 10005
03 int n, m, wn;
04 int fa[N], c[N], d[N];
05 int v[N], w[N], an, dp[N];
06 int find(int x) {
07     return fa[x] == x ? x : (fa[x] = find(fa[x]));
08 }
09 void merge(int i, int j) {
10     int x = find(i), y = find(j);
11     if (x != y) {
12         fa[x] = y;
13         __①__
14         d[y] += d[x];
15     }
16 }
17 int main() {
18     int f, t;
19     std::cin >> n >> m >> wn;
20     for (int i = 1; i <= n; ++i) fa[i] = i;
21     for (int i = 1; i <= n; ++i)
22         std::cin >> c[i] >> d[i];
23     for (int i = 1; i <= m; ++i) {
24         std::cin >> f >> t;
25         __②__
26     }
27     for (int i = 1; i <= n; ++i) {
28         if (fa[i] == i) {
29             v[++an] = d[i];
30             __③__
31         }
32     }
33     for (int i = 1; i <= an; ++i)
34         for (int j = wn; j >= w[i]; --j)
35             dp[j] = std::max(__④__, dp[j - w[i]] + v[i]);
36     std::cout << __⑤__;
37     return 0;
```

```
38  }
```

56. ① 处应该填（　　）。

A. c[y] -= c[x];

B. c[y] += c[x];

C. d[x] -= d[y];

D. d[x] += d[y];

57. ② 处应该填（　　）。

A. merge(f, i);

B. merge(i, t);

C. merge(t, an);

D. merge(f, t);

58. ③ 处应该填（　　）。

A. fa[an] = c[i];

B. d[an] = c[i];

C. w[an] = c[i];

D. dp[an] = c[i];

59. ④ 处应该填（　　）。

A. dp[i]

B. dp[j]

C. dp[w[i]]

D. dp[j - w[i]]

60. ⑤ 处应该填（　　）。

A. dp[0]

B. dp[1]

C. dp[an]

D. dp[wn]