一、单选

2020年专业基础能力测评

A. Multiple-Choice - 1 2 | Multiple-Choice - M 1 | Fill-in-Blank -

2-1下列关于指针的运算中，哪一个是非法的？ (3 point(s))

- A.两个指针在一定条件下，可以进行相等或不等的运算。
- B.可以有一个空指针赋值给某个指针。
- C.一个指针可以加上一个整数。
- D.两个指针在一定条件下，可以相加。 **指针相加，地址空间表示上，没有意义**

Author **葛亮**

Organization 重庆大学

2-1 Accepted (3 point(s))

A. Multiple-Choice - 1 2 | Multiple-Choice - M 1 | Fill-in-Blank -

2-2下列种类的函数中，哪一种不是类的成员函数？ (3 point(s))

- A.构造函数
- B.析构函数
- C.友元函数 √ 只需声明为友元函数
- D.拷贝构造函数

Author 葛亮

Organization 重庆大学

2-2 Accepted (3 point(s))

二、多选

A. Multiple-Choice - 1 2 | Multiple-Choice - M 1 | Fill-in-Blank -

Selecting more answer options than the number of correct answer options results in zero, while selecting less results in half score.

3-1面向对象程序设计的三大特性是什么？ (3 point(s))

- [x] A.封装
- [x] B.继承
- [x] C.多态
- [ ] D.函数

Author 葛亮

Organization 重庆大学

3-1 Accepted (3 point(s))

三、程序填空

5-1

下列程序实现冒泡排序，请在空缺处填上正确的代码。

```
#include <iostream>
#include<vector>
using namespace std;
void bubbleSort(vector<int>&arr)
{
    int n=arr.size();
    for(int i = 0; i <
```

n-1

2 point(s)

```
; i++)
    {
        for(int j = 0; j <
```

n-i

3 point(s)

```
; j++)
        {
            if(arr[j] > arr[j+1])
                swap(arr[j],arr[j+1]);
        }
    }
}
int main()
{
    int n;
    cin>>n;
    vector<int> arr(n);
    for(int i=0;i<n;i++)
        cin>>arr[i];
    bubbleSort(arr);
    for(int i=0;i<n;i++)
        cout<<arr[i]<<endl;
    return 0;
}
```

5-1 Accepted(5 point(s))

5-2

汉诺塔是经典的递归问题。该问题描述为：有三根相邻的柱子，标号为 A、B、C，A 柱子上从下到上按金字塔状叠放着 n 个不同大小的圆盘，要把所有盘子一个一个移动到柱子 C 上，并且每次移动同一根柱子上都不能出现大盘子在小盘子上方，求移动的过程。

下列程序是汉诺塔问题的求解程序，其中函数 move(n,A,B,C)表示将 n 个圆盘从 A 借助 B 搬到 C 中，请补全程序中的空白部分。

```
#include <iostream>
using namespace std;

void move(int n,char A,char B,char C);
int step;
int main()
{
    int n;
    cin>>n;
    move(n,'A','B','C');
    return 0;
}
void move(int n,char A,char B,char C)
{
    if(
```

n==1

1 point(s)

```
)
    {
        step++;
        cout<<"["<<step<<"]move 1# from "<<A<<" to "<<C<<endl;
    }
    else
    {
```

move(n-1, A, C, B)

2 point(s)

```
;
        cout<<"["<<step<<"]move "<<n<<"# from "<<A<<" to "<<B<<endl;
        step++;
```

move(n-1,B,C,A)

n ;

2 point(s)

```
;
    }
}
```

5-2 Partially Accepted (3 point(s))

5-3

下列程序演示了 C++的多态性，请补充空白处的代码，使其能够正确运行。

```
#include <iostream>
```

```
using namespace std;
class Shape{
    private:
        int size;
    public:
        Shape(int _size):size(_size){
            cout<<"Shape::Shape(int _size)"<<endl;
        }
```

virtual

虚函数 → 多态

1 point(s)

```
 void draw(){
            cout<<"Shape::draw()"<<endl;
        }
        virtual ~Shape()
        {
            cout<<"Shape::~Shape()"<<endl;
        }
};
class Circle:
```

public Shape

继承

1 point(s)

```
{
    public:
        Circle(int _size):
```

size(_size)

初始化列表

1 point(s)

```
        {

        }
        void draw(){
            cout<<"Circle::draw()"<<endl;
        }
        ~Circle()
        {
            cout<<"Circle::~Circle()"<<endl;
        }
};
int main()
{
    int size;
```

```
    cin>>size;
```

Shape

父类

1 point(s)

```
circle=new Circle(size);
    circle->draw();
```

~circle

析构

1 point(s)

```
;
    return 0;
}
```

5-3 Partially Accepted(2 point(s))

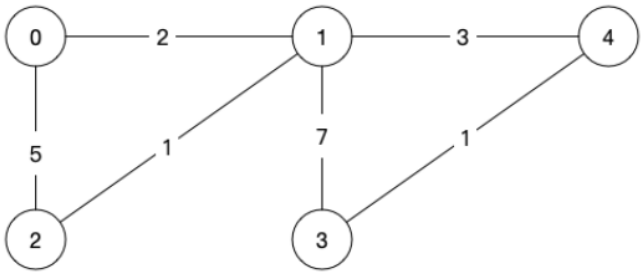
5-4下列程序是Dijkstra算法的C++实现，程序的输入格式为：

```
n  //顶点数
m //边数
a1 b1 c1 //顶点号 顶点号 边权
a2 b2 c2 //顶点号 顶点号 边权
a3 b3 c3 //顶点号 顶点号 边权
......
```

例如：

```
5
5
0 1 2
0 2 5
1 4 3
1 2 1
3 4 1
```

表示如下的无向图：



经过程序的计算，输出顶点0到其他顶点的最短路径长度为：

```
0 //顶点0到顶点0
2 //顶点0到顶点1
3 //顶点0到顶点2
6  //顶点0到顶点3
5  //顶点0到顶点4
```

请补充空白处的程序代码，使程序能够正确运行。

```
#include <iostream>
using namespace std;
#define INF 99999999
void dijkstra();
int e[10][10];  //邻接矩阵
int vis[10];    //访问标记
int dis[10];    //v0 到 vi 的最短距离
int n; //n 个顶点
int m; //m 条边
int main()
{
    cin >> n;
    // 初始邻接矩阵
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (i == j)
            {
```

e[i][j]=0

```
e[i][j]=
```

2 point(s)

```
;
            }
            else
            {
```

```
                e[i][j] = INF;
            }
        }
    }
    // 初始化访问标记
    for(int i=0;i<n;i++)
```

vis[i]=0

2 point(s)

```
;
    cin>>m;
    //设置边权
    for (int i = 0; i < m; i++)
    {
        int a, b, c;
        cin >> a >> b >> c;
        e[a][b] = c;
```

e[b][a]=c 无向图

2 point(s)

```
;
    }
    //初始化最短距离
    for (int i = 0; i < n; i++)
    {
        dis[i] = e[0][i];
    }
    vis[0] = 1;

    dijkstra(); //求最短路径
    //输出最短路径
    for (int i = 0; i < n; i++)
    {
        cout << dis[i]<<endl;
    }
    return 0;
}
void dijkstra()
{

    for (int i = 0; i < n - 1; i++)
    {
```

到n-1个点，只需要循环n-1次，即可找到最短路径

```
        int min1 = INF;
        int u=0;
        // 寻找权值最小的点 u
        for (int j = 0; j < n; j++)
        {
            if (vis[j] == 0 && dis[j] < min1)
            {
                min1 = dis[j];
                u = j;
            }
        }

        vis[u] = 1;

        for (int v = 0; v < n; v++)
        {
            if (e[u][v] < INF)
            {

                if (
```

dis[v]>dis[u]+e[u][

dis[v] > dis[u]+e[u][v] && !vis[u]

2 point(s)

```
)
                {
                    dis[v] = dis[u] + e[u][v];
                }
            }
        }
    }
}
```

5-5

下列 C++程序实现了一个顺序表，请补充空白处的程序代码，使其能够正确运行。

```
#include <iostream>
using namespace std;
template<typename ElemType>
class ArrayList
{
private:
    ElemType *elems;  //数据元素
    int sizeOfElems;  //数据个数
    int sizeOfAllocatedMemory;  //分配内存大小
```

```
    static const int INC=10;//内存增量
    void reallocate(); //重新分配内存
public:
    ArrayList(int size=0);  //构造函数
    ArrayList(const ArrayList<ElemType>& list); //拷贝构造函数
    ~ArrayList();  //析构函数
    int size() const; //返回数据个数
    void add(int index, ElemType e); //插入数据
    void add(ElemType e); //插入数据
    void remove(int index); //移除数据
    void remove(); //移除数据
    void clear(); //清除全部数据
    ElemType& operator[](int index); //返回数据
    ArrayList<ElemType>& operator=(const ArrayList<ElemType>& list); //=运算符重载
    friend
```

operator<<

2 point(s)

```
(ostream& os, const ArrayList<ElemType>& list) //<<运算符重载
    {
        for(int i=0;i<list.size();i++)
            cout<<list.elems[i]<<"\t";
        return os;
    }
};
template<typename ElemType>
ArrayList<ElemType>::ArrayList(int size)
{
    elems=new ElemType[size];
    sizeOfElems=0;
    sizeOfAllocatedMemory=size;
}
template<typename ElemType>
ArrayList<ElemType>::ArrayList( const ArrayList<ElemType>& list )
{
    elems=new ElemType[list.sizeOfAllocatedMemory];
    sizeOfAllocatedMemory=list.sizeOfAllocatedMemory;
    sizeOfElems=list.sizeOfElems;
    for(int i=0;i<list.sizeOfElems;i++)
```

new动态申请空间

elems[i]=list.elems

elems[i]=list.elems[i]

1 point(s)

```
;
}
```

```
template<typename ElemType>
ArrayList<ElemType>::~ArrayList()
{
    delete[] elems;
}
template<typename ElemType>
int ArrayList<ElemType>::size() const
{
    return sizeOfElems;
}
template<typename ElemType>
void ArrayList<ElemType>::reallocate()
{
    int newSizeOfAllocatedMemory=sizeOfAllocatedMemory+INC;
    ElemType *newElems=new ElemType[newSizeOfAllocatedMemory];
    for(int i=0;i<sizeOfElems;i++)
    {
        newElems[i]=elems[i];
    }
    delete []elems;
    elems=newElems;
    sizeOfAllocatedMemory=newSizeOfAllocatedMemory;
}
template<typename ElemType>
void ArrayList<ElemType>::add( int index, ElemType e )
{
    if (index>sizeOfElems) {
        throw string("index out of bounds!");
    }
    int newSizeOfArray=sizeOfElems+1;
    if(sizeOfAllocatedMemory<newSizeOfArray)
    {
        reallocate();
    }
    for(int i=sizeOfElems-1;i>=index;i--)
    {
        elems[i+1]=elems[i];
    }
    elems[index]=e;
    sizeOfElems=newSizeOfArray;
}
template<typename ElemType>
void ArrayList<ElemType>::add( ElemType e )
{
    add(this->sizeOfElems,e);
```

```
}
template<typename ElemType>
void ArrayList<ElemType>::remove( int index )
{
    int newSizeOfElems=sizeOfElems-1;
    if(index>newSizeOfElems || index<0)
    {
        throw string("index out of bounds!");
    }
    if(sizeOfElems==0)
    {
        throw string("There is no element to remove!");
    }
    for(int i=index;i<newSizeOfElems;i++)
    {
        elems[i]=elems[i+1];

    }
    sizeOfElems=newSizeOfElems;
}
template<typename ElemType>
void ArrayList<ElemType>::remove()
{
    remove(0);
}
template<typename ElemType>
void ArrayList<ElemType>::clear()
{
```

~ArrayList()

1 point(s)

```
;
}
template<typename ElemType>
ElemType& ArrayList<ElemType>::operator[]( int index )
{
    if(index>sizeOfElems-1 || index<0)
    {
        throw string("Index out of bounds");
    }
    return
```

elems[index]

2 point(s)

```
;
}
template<typename ElemType>
ArrayList<ElemType>& ArrayList<ElemType>::operator=( const ArrayList<ElemType>& list )
{
    delete []elems;
    elems=new ElemType[list.sizeOfAllocatedMemory];
    sizeOfAllocatedMemory=list.sizeOfAllocatedMemory;
```

sizeOfElems=list.si sizeOfElems=list.sizeOfElems

2 point(s)

```
;
    for(int i=0;i<list.sizeOfElems;i++)
        elems[i]=list.elems[i];
    return *this;
}


int main(int argc, const char * argv[]) {
    try
    {
        ArrayList<int> l1;
        l1.add(0,10);
        l1.add(0,20);
        l1.add(0,30);
        l1.add(40);
        cout<<"l1:"<<l1<<endl;
        ArrayList<int> l2(l1);
        l2.remove(1);
        cout<<"l2:"<<l2<<endl;
        l2.remove();
        cout<<"l2:"<<l2<<endl;
        ArrayList<int> l3=l2;
        cout<<"l3:"<<l3<<endl;
        cout<<"The size of l3 is "<<l3.size()<<endl;
        l3[0]=1000;
        cout<<"l3:"<<l3<<endl;
        l3.clear();
        cout<<"l3:"<<l3<<endl;
    }
    catch (string e)
    {
```

```
        cout<<"Exception: "<<e<<endl;
    }
    return 0;
}
```

四、编程题

返回

7-1 字符串转换 (15 point(s))

请按如下要求编写程序：

1. 输入第1行是数字n，表示接下来将有n行字符串。
2. 输入第2行到第n+1行的每一行是一个字符串。
3. 将每行字符串中的偶数位删除，输出新的字符串。
4. 若字符串中不包含任何字母（A-Z或者a-z），则保持原字符串不变。

输入格式:

```
n
第1行字符串
第2行字符串
.....
第n行字符串
```

输出格式:

```
转换后的第1行字符串
转换后的第2行字符串
.....
转换后的第n行字符串
```

输入样例:

```
3
abc
efgh
1234
```
结尾无空行

输出样例:

```
ac
eg
1234
```
结尾无空行

返回

7-2 最优装车 (15 point(s))

老王是一名仓库装车员，他的工作是将货物搬运到货车上。为了使运输效益最大化，运输公司的老板希望各辆车所装货物的重量差值越小越好。老王面临的问题是一个NP问题，幸好目前只有2辆车，4件货物。聪明的老王知道可以用穷举法来解决这个问题，并且只需要穷举14种情况。请你写一个程序，帮老王计算一下，最优的货物装车差值是多少？

例如如果4件货物的重量分别是[4,5,3,1]，那么一辆车装[4,3]，另一辆车装[5,1]，两辆车装货重量的差值是1，这也是最优的货物装车差值。

输入格式:

4个空格分割的数字，代表4件货物的重量。

输出格式:

1个数字，最优的货物装车差值

直接暴力出奇迹

输入样例:

```
4 5 3 1
```
结尾无空行

输出样例:

```
1
```
结尾无空行

返回

7-2 用BMI判断胖瘦 (15 point(s))

体质指数 (Body Mass Index，简称BMI)，是国际最常用来量度体重与身高比例的工具。它利用身高和体重之间的比例去衡量一个人是否过瘦或过肥。用体重$weight$（单位kg）和身高$height$（单位m）计算BMI的公式为：

$BMI = weight/height^2$

例如体重$70kg$，身高$1.6m$，则$BMI = 70/1.6^2 = 27.3$。

亚洲成年人用以下的指引判断体重是否正常：

BMI	类别
< 18.5	过轻
18.5 - 23.9	正常
> 23.9	超重

输入格式:

$weight$ $height$，用空格分割的体重和身高。

输出格式:

若**体重过轻**，输出-1；

若**体重正常**，输出0;

若**体重超重**，输出1。

输入样例:

```
70 1.6
```
结尾无空行

输出样例:

```
1
```
结尾无空行

返回

7-3 0-1背包问题 (15 point(s))

N件物品，第i件物品的重量和价值分别为w_i和$v_i(1 \le i \le N)$，背包容量为W，选择部分物品装入背包，在物品总重量不超过背包容量的条件下，求装入的物品总价值最大的最优装法。

输入格式:

第一行给出正整数$N(\le 100)$和$W(\le 1000)$。接下来N行数据，每行给出两个正整数，第i行的数据对应第i件物品的重量w_i和价值v_i $(0 < w_i, v_i \le 100)$。

很正常的0-1背包

输出格式:

输出装入物品后获得的最大价值。

输入样例:

在这里给出一组输入。例如：

```
4 4
2 4
3 5
1 4
2 5
```
结尾无空行

输出样例:

在这里给出相应的输出。例如：

```
9
```
结尾无空行

返回

7-3 解一元二次方程 (15 point(s))

一元二次方程$ax^2 + bx + c = 0$的求根公式为：

$x_1 = \frac{-b-\sqrt{\Delta}}{2a}, x_2 = \frac{-b+\sqrt{\Delta}}{2a}$

其中：

$\Delta = b^2 - 4ac$

请利用上述求根公式，求出一元二次方程的解。

注意：题目中不会出现$\Delta \le 0$的情况。

输入格式:

a b c，一元二次方程的系数，以空格分割。

输出格式:

x_1 x_2，一元二次方程的解，以空格分割，小数点后保留两位有效数字（四舍五入），并且$x_1 \le x_2$

输入样例:

```
1 -1 -6
```
结尾无空行

输出样例:

在这里给出相应的输出。例如：

```
-2.00 3.00
```
结尾无空行

返回

7-4 最小生成树 (15 point(s))

对有n个顶点、m条边的无向图，计算其最小生成树并输出树的权重，即构成最小生成树的所有边的权重值总和。如果无向图连通，计算结果是一个含$(n-1)$条边的最小生成树；如果图不连通，则依次计算各个连通子图的最小生成树，再将所有树的权重相加。

输入格式:

第一行给出正整数$n(\leq 1000)$和$m(\leq 10000)$，分别表示无向图的顶点和边的数量。接下来m行数据，每行给出三个非负整数，其中第i行$(1 \leq i \leq m)$的数据u_i, v_i, w_i表示图的第i条边$(u_i,\ v_i)$和边的权重$w_i(0 \leq u_i < v_i < n,\ 0 < w_i \leq 100)$。

输出格式:

在一行中输出两个整数，用空格分开。第一个整数表示无向图的最小生成树的权重，第二个整数是图中连通子图的数量（如果图连通，则输出1）。

输入样例1:

在这里给出一组输入。例如：

```
5 5
0 1 1
1 2 1
2 3 1
3 4 1
0 4 2
```

ⓘ 结尾无空行

输出样例1:

在这里给出相应的输出。例如：

```
4 1
```

ⓘ 结尾无空行

输入样例2:

在这里给出一组输入。例如：

```
5 2
0 1 1
2 3 1
```

ⓘ 结尾无空行

输出样例2:

在这里给出相应的输出。例如：

```
2 3
```

ⓘ 结尾无空行