2020 年程序员考试下午真题

1、

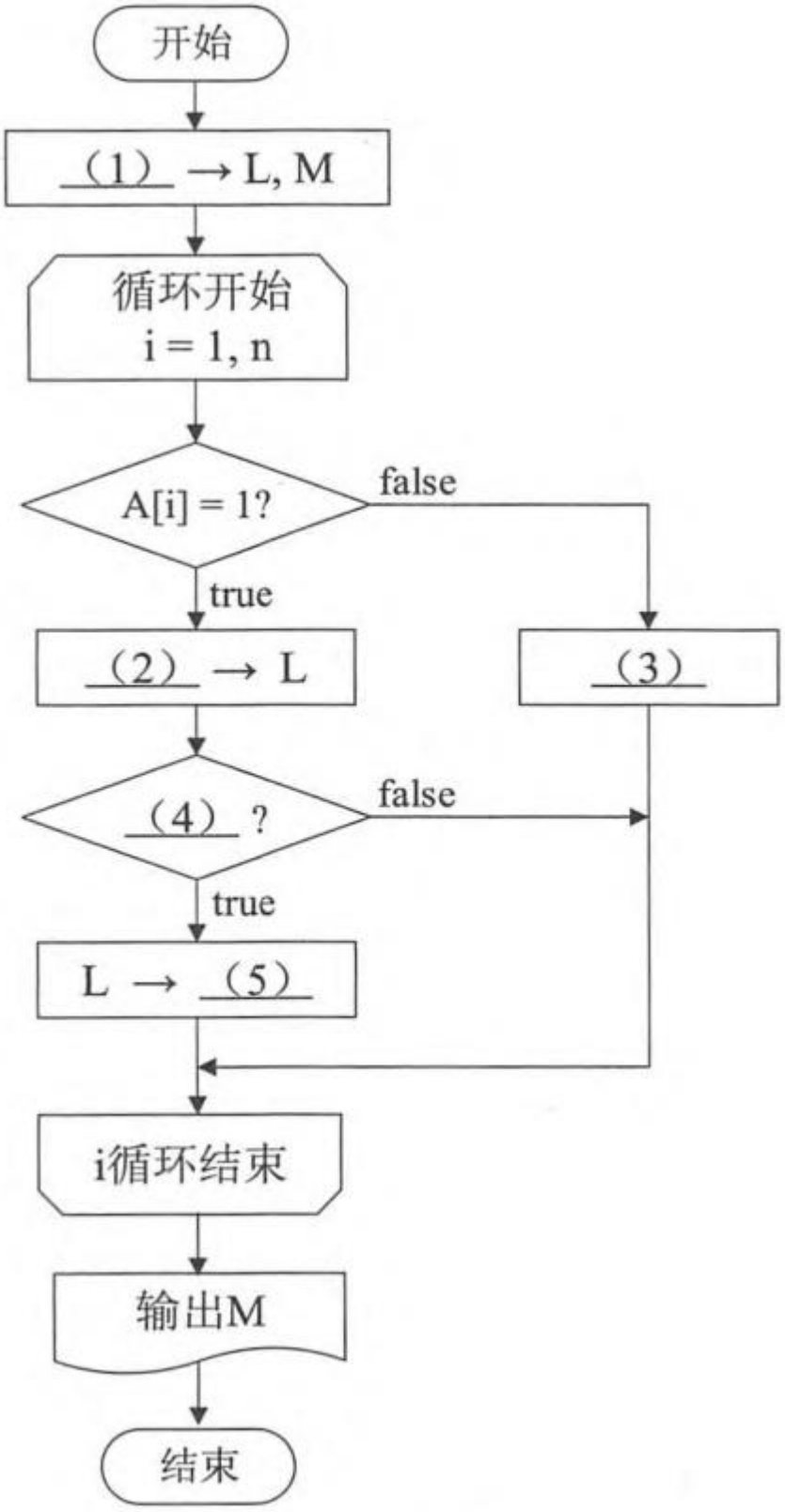
阅读以下说明和流程图，填写流程图中的空缺，将解答填入答题纸的对应栏内。

【说明】

下面流程图所示算法的功能是：在一个二进制位串中，求出连续的“1”构成的所有子串的最大长度 M。例如，对于二进制位串 0100111011110，M=4。

该算法中，将长度为 n 的二进制位串的各位数字，按照从左到右的顺序依次存放在数组 A[1..n]。在对各个二进制位扫描的过程中，变量 L 动态地记录连续“1”的个数。

【流程图】



注：循环开始框内应给出循环控制变量的初值和终值，默认递增值为 1。

格式为：循环控制变量=初值，终值 [,递增值]

2、

阅读以下说明和 C 代码，填补 C 代码中的空缺，将解答写在答题纸的对应栏内。

【说明】

函数 cubeRoot(x)的功能是用下面的迭代公式求解 x 的立方根的近似值：

$$x_{n+1}=\frac{1}{3}(2x_n+x/x_n^{\,2}),\quad \text{精度要求为}\left|(x_{n+1}-x_n)/x_n\right|<10^{-6}$$

在 main()函数中，调用 cubeRoot(x)计算区间[–8.0, 8.0]中各浮点数（间隔 0.1）的立方根。

【C 代码】

```
#include<stdio.h>
#include<math.h>
double cubeRoot(double x) {
    double  x1, x2 = x;

    if( __(1)__ )return 0.0;
    do {
       x1 = __(2)__;
       x2 = (2.0 * x1 + __(3)__ ) / 3.0;
    }while(fabs( __(4)__ ) >= 1e-6);
   return x2;
}
int main()
{
   double x;
   for(x = -8.0; x <= 8.0; __(5)__)
     printf("cube_root(%.1f)=%.4f\n", x,cubeRoot(x));

   return 0;
}
```

3、

阅读以下说明和 C 代码，填补 C 代码中的空缺，将解答写在答题纸的对应栏内。

【说明】

下面程序中，函数 convertion(char *p)的功能是通过调用本程序中定义的函数，将 p 所指示字符串中的字母和数字字符按如下约定处理：

（1）大写字母转换为小写字母；

（2）小写字母转换为大写字母；

（3）数字字符转换为其伙伴字符（当两个十进制数字相加为 9 时，这两个十进制数字对应的数字字符互为伙伴字符）。例如，字符'2'的伙伴字符为'7'、'8'的伙伴字符为'1'、'0'的伙伴字符为'9'等。

【C 代码】

```
#include<stdio.h>
int isUpper(char c) {                    //判断 c 表示的字符是否为大写字母
    return (c>='A' && c<='Z');
}
int isLower(char c) {                    //判断 c 表示的字符是否为小写字母
    return (c>='a' && c<='z');
}
int isDigit(char c) {                    //判断 c 表示的字符是否为数字字符
    return (c>='0' && c<='9');
}
void toUpper(char *c) {                  //将 c 指向的小写字母转换为大写字母
    *c = *c - 'a' + 'A';
}
void toLower(char *c) {                  //将 c 指向的大写字母转换为小写字母
    *c = *c - 'A' + 'a';
}
void cDigit(char *c) {                   //将 c 指向的数字字符转换为其伙伴字符
    *c = 9 - ( __(1)__ ) + '0';
}
void convertion(char *p) {
    while (*p) {
        if ( __(2)__ ) {
            toLower(p);
        }
        else if ( __(3)__ ) {
            toUpper(p);
        }
        else if ( __(4)__ ) {
            cDigit(p);
        }
        __(5)__;
    }
}
int main() {
char str[81] = "Aidf3F4";

    printf("%s\n",str);                  //输出为 Aidf3F4
    convertion(str);
    printf("%s\n",str);                  //输出为 aIDF6f5
    return 0;
}
```

4、

阅读以下说明和 C 代码，填补 C 代码中的空缺，将解答写在答题纸的对应栏内。

【说明】

函数 createList(int a[ ],int n)根据数组 a 的前 n 个元素创建对应的单循环链表，并返回链表的尾结点指针。例如，根据数组 int a[] = {25,12,39}创建的单循环链表如图 4-1 所示。

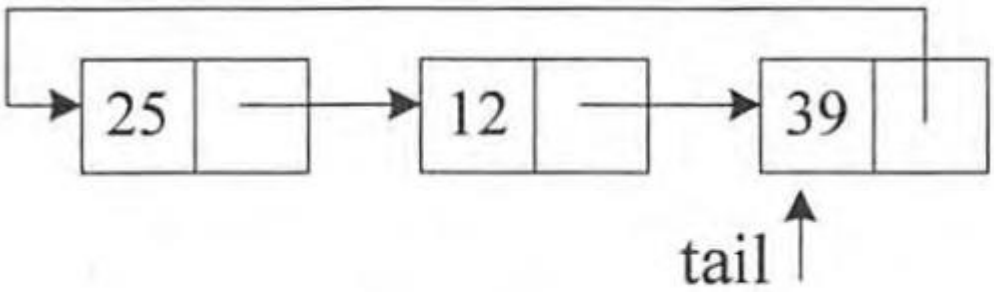


图 4-1

函数 display(NodePtr tail)的功能是从表头元素开始，依次输出单循环链表中结点数据域的值。

链表的结点类型定义如下：

```
typedef struct Node *NodePtr;
struct Node{
    int key;
    NodePtr  next;
};
```

【C 代码】

```
NodePtr createList(int a[], int n)  { //根据数组 a 的前 n 个元素创建单循环链表
    NodePtr tail = NULL,  p;

    if (n<1) return NULL;
    p = (NodePtr)malloc(sizeof(struct Node));   //创建第一个结点
    if (!p) return tail;

    p->key = a[0];
    p->next = p;
     (1)    = p;

    for(int i=1; i<n; i++) {                          //创建剩余的 n-1 个结点
        p = (NodePtr)malloc(sizeof(struct Node));
        if (!p) break;
         (2)    = a[i];
         (3)    = tail->next;
        tail->next = p;
        tail = p;
    }
    return tail;
}
void display(NodePtr tail) {
//从表头元素开始，依次输出单循环链表中结点数据域的值
     if(  (4)   )return;
     NodePtr p = tail->next;
     do{
         printf("%d\t", p->key);
         p =  (5)  ;
     }while( p != tail->next );
}

int main() {
    int a[] = {25,12,39};
    NodePtr tail;
   tail = createList(a,3);
   display(tail);   //输出 25 12 39
   return 0;
}
```

5、

阅读以下说明和 Java 代码，填写 Java 代码中的空缺，将解答写入答题纸的对应栏内。

【说明】

在线购物系统需提供订单打印功能，相关类及关系如图 5-1 所示，其中类 Order 能够完成打印订单内容的功能，类 HeadDecorator 与 FootDecorator 分别完成打印订单的抬头和脚注的功能。

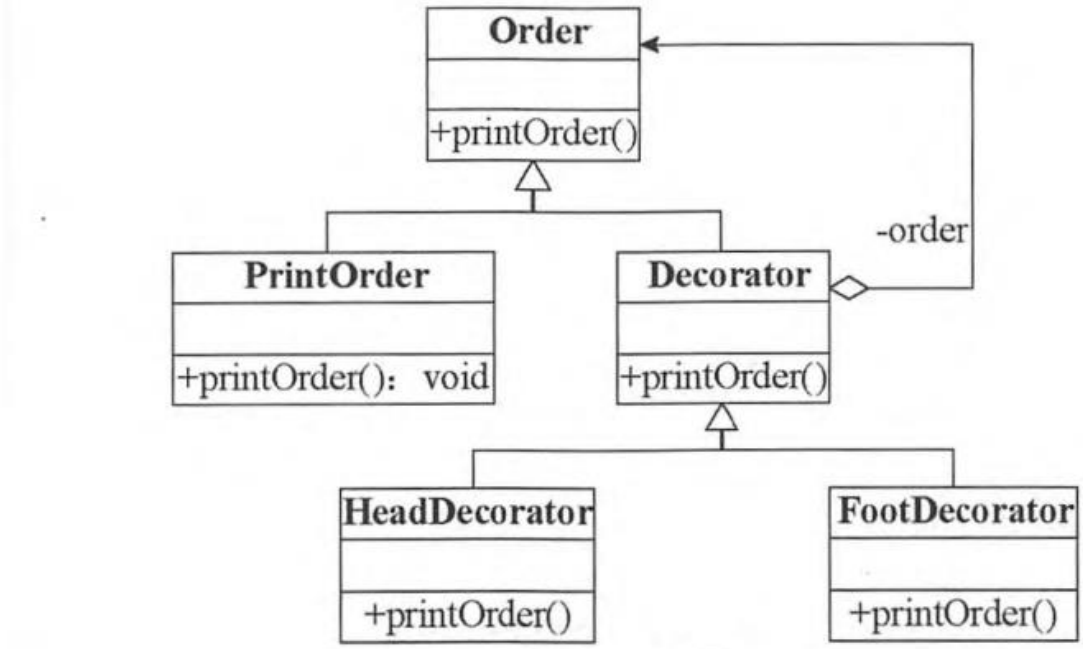


图 5-1 类图

下面的 Java 代码应实现上述设计，其执行结果如下：

```
订单抬头!
订单内容!
订单脚注!
------------------------
订单抬头!
订单脚注!
```

【Java 代码】

```
class Order {
    public void printOrder() { System.out.println("订单内容!");}
}

class Decorator ___(1)___ Order {
    private Order order;

    public Decorator(Order order){ ___(2)___ = order; }
    public void printOrder(){
        if(order != null)
            order.printOrder();
    }
}
class HeadDecorator extends Decorator {
    public HeadDecorator(Order order) { ___(3)___;}
    public void printOrder() {
        System.out.println("订单抬头! ");
        super.printOrder();
    }
}
class FootDecorator extends Decorator{
    public FootDecorator(Order order) { ___(4)___; }
    public void printOrder() {
        ___(5)___;
        System.out.println("订单脚注! ");
    }
}
class PrintOrder ___(6)___ Order {
    private Order order;

    public PrintOrder(Order order) {___(7)___ = order;}
    public void printOrder() {
        FootDecorator foot = new FootDecorator(order);
        HeadDecorator head = new HeadDecorator(foot);
        head.printOrder();
        System.out.println("-----------------------");
        FootDecorator foot1 = new FootDecorator(null);
        HeadDecorator head1 = new HeadDecorator(foot1);
        head1.printOrder();
    }
    public static void main(String[] args)  {
        Order order = new Order();
        PrintOrder print = new PrintOrder(order);
        print.printOrder();
    }
}
```

6、

阅读下列说明和 C++代码，填写 C++代码中的空缺，将解答写入答题纸的对应栏内。

【说明】

在线购物系统需提供订单打印功能，相关类及关系如图 6-1 所示，其中类 Order 能够完成打印订单内容的功能，类 HeadDecorator 与 FootDecorator 分别完成打印订单的抬头和脚注的功能。

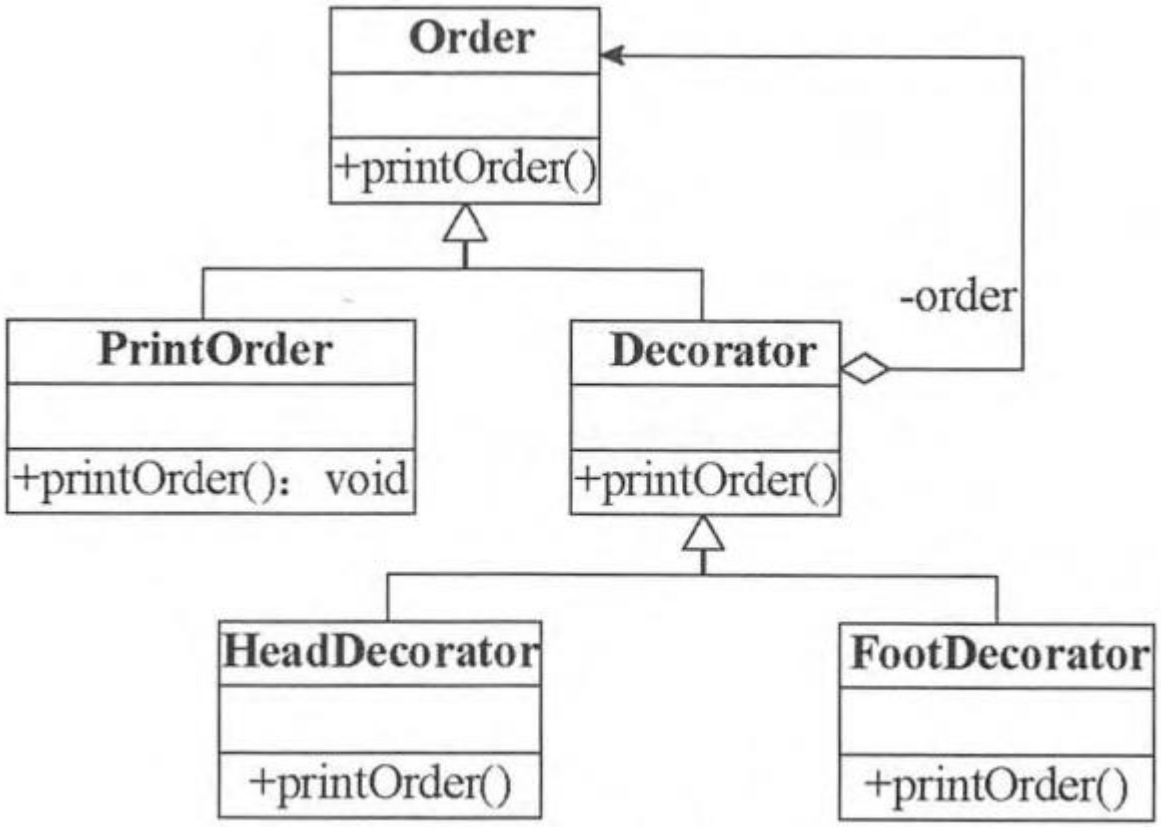


图 6-1 类图

下面的 C++代码应实现上述设计，其执行结果如下：

```
订单抬头!
订单内容!
订单脚注!
------------------------
订单抬头!
订单脚注!
```

【C++代码】

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

class Order {
public:
    virtual void printOrder() { cout << "订单内容!" << endl;}
};

class Decorator ___(1)___ {
private:    Order *order;
public:
    Decorator(Order *order){___(2)___ = order; }
    void printOrder(){
        if(order != NULL)
            order->printOrder();
    }
};
class HeadDecorator : public Decorator {
public:
    HeadDecorator(Order *order):___(3)___ { }
    void printOrder() {
        cout << "订单抬头! " << endl;
        Decorator::printOrder();
     }
 };
 class FootDecorator :public Decorator{
  public:
     FootDecorator(Order *order): ___(4)___ { }
     void printOrder() {
         ___(5)___;
         cout << "订单脚注! " << endl;
     }
 };
 class PrintOrder ___(6)___ {
 private:    Order *order;
   public:
       PrintOrder(Order *order){___(7)___ = order;}
       void printOrder() {
           FootDecorator *foot = new FootDecorator(order);
           HeadDecorator *head = new HeadDecorator(foot);
           head->printOrder();
           cout << "-----------------------"<< endl;
           FootDecorator foot1(NULL);
           HeadDecorator head1(&foot1);
           head1.printOrder();
       }
   };

   int main() {
       Order *order = new Order();
       Order *print = new PrintOrder(order);
```