华北计算技术研究所 2004 年专业课试题

要求：1、答案必须写在答题纸上，标明题号；

2、答卷要字迹清楚，语义确切；

3、所有计算要求给出计算过程。

1.（10 分）

（1）以 n、a_i(i=0,1, . . . , n)、x_0 作为输入，为了进行一元 n 次多项式 $P_n(x)=a_0x^n+a_1x^{n-1}+a_2x^{n-2}+\ldots+a_{n-1}x+a_n$ 在 x_0 点的值 $P_n(x_0)$的计算，请给出你认为效率最好的算法。

（2）给出上述算法的基本操作、基本操作执行次数和时间复杂度。

2.（10 分）

设有三对角矩阵(a_{ij})nxn，将其三条对角线上的元素逐行地存于数组 B[3n-2]中，使得 B[k]=a_{ij}，求：

（1）用 i，j 表示 k 的下标变换公式；

（2）用 k 表示 i，j 的下标变换公式。

3.（10 分）

（1）已知一棵二叉树的先序序列为 EBADCFHGIKJ 和中序序列为 ABCDEFGHIJK，请画出该树，并给出计算或推理过程。

（2）已知一棵二叉树的中序序列为 DCBGEAHFIJK 和后序序列为 DCEGBFHKJIA，请画出该树，并给出计算或推理过程。

4.（15 分）

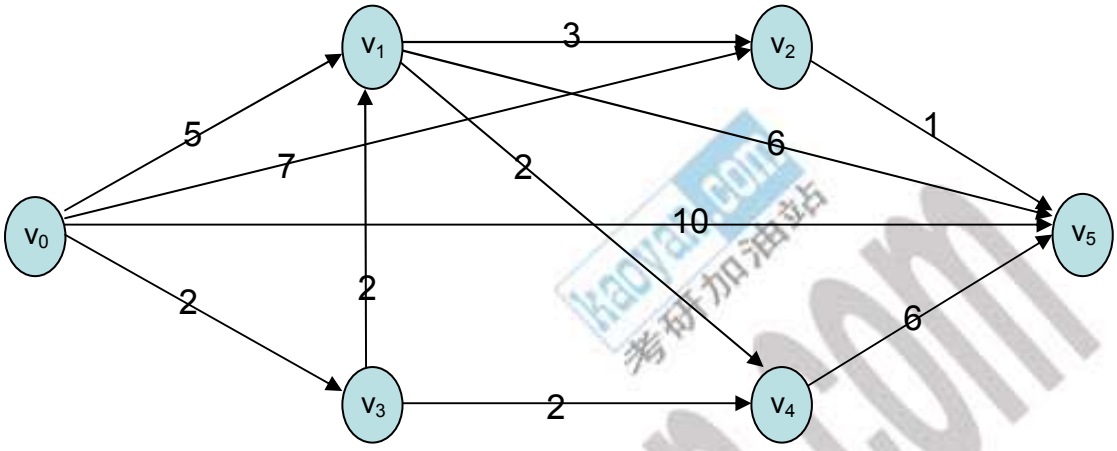
某人自下往上走完一个 N 级的台阶，每步只能走一级或两级台阶：

（1）给出能够计算出上述台阶所有走法的递归算法。

（2）以 C 或 C^{++}实现上述算法。

5．（20 分）

下图是一个有向图，其中每条弧段上的数字表示该弧段的权值。请用 Dijkstra 算法计算 v_0 到各点的最短路径及路径的长度（要求给出计算过程）。



6．（30 分）

已知如下所示长度为 12 的表

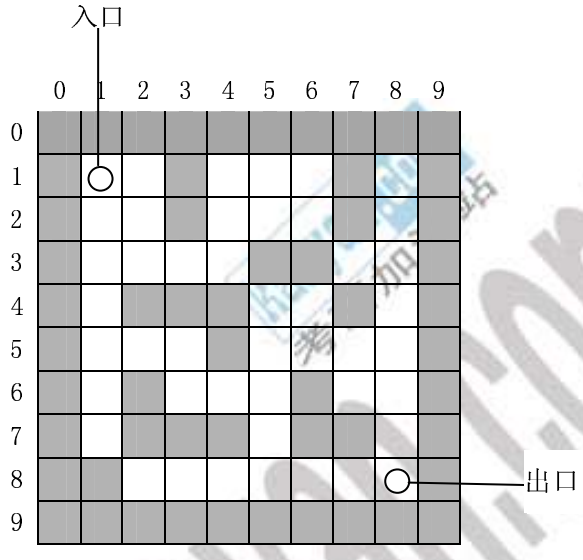
（Jan，Feb，Mar，Apr，May，June，July，Aug，Sep，Oct，Nov，Dec）

（1）试按表中元素的顺序依次插入一棵初始为空的二叉排序树，画出插入完成之后的二叉排序树，并求其在等概率情况下查找成功的平均查找长度。

（2）若对表中元素先进行排序构成有序表，求在等概率情况下对此有序表进行折半查找时查找成功的平均查找长度。

（3）请按表中元素的顺序构造一棵平衡二叉排序树，并求其在等概率情况下查找成功的平均查找长度。

7.（25 分）如图所示的方块图表表示一个迷宫。图中的每个白方块表示为通道，黑方块为墙。请在①、②、③处填充必要的 C 语言代码，完成下面求从迷宫入口到出口路径的程序。




```
#define STACK_INCREMENT 10                  //栈每次增加的大小
#define OK true
#define ERROR false
#define MAZEWIDTH 10                        //迷宫的X方向大小
#define MAZEHEIGHT 10                       //迷宫的Y方向大小

 //坐标位置状态,0----没有走过，1----走过了，2----不通,3----是墙壁
enum status {
    NOT_PASSED,   //没有走过该通道块
    PASSED,       //该通道块已经走过了
    NOT_THROUGH,  //不通
    IS_WALL       //是墙壁
};

typedef struct postype
{
    int x;  //横坐标
    int y;  //纵坐标
}PosType;

typedef struct selemtype
{
```

```
    int ord;         //通道块在路经中的"序号"
    PosType seat;    //通道块在迷宫中的"坐标位置"
    int di;          //从此通道块走向下一个通道块的"方向"
                     //1----东面，2---南面，3---西面，4---北面
}SElemType;     //栈的元素类型

typedef struct
{
    SElemType *base;    //栈底
    SElemType *top;     //栈顶
    int stacksize;      //栈大小
}SqStack;               //栈结构

//构造一个空栈
bool InitStack(SqStack &S)
{
    S.base = (SElemType *)malloc(STACK_INIT_SIZE * sizeof(SElemType));

    if(!S.base) return ERROR;
    S.top = S.base;
    S.stacksize = STACK_INIT_SIZE;
    return OK;
}

//判断栈是否为空
bool StackEmpty(SqStack S)
{
    if(S.base == S.top)
        return true;
    else
        return false;
}

//插入元素E为新的栈顶元素
bool Push(SqStack &S,SElemType e)
{
```

```
}
//若栈不空，则删除S的栈顶元素，用E返回其值，并返回OK，否则返回ERROR
bool Pop(SqStack &S,SElemType &e)
{
```

②

```
}

//能否通过curpos位置的通道块
bool Pass(int *maze,PosType curpos)
{
    if(maze[curpos.x * MAZEWIDTH + curpos.y] == NOT_PASSED)
        //当前的还没有走过
        return true;
    else
        return false;
}

//maze是个二维的迷宫矩阵
//curpos是当前的通道块
//di是下一个通道块的方向
PosType NextPos(int *maze,PosType curpos,int di)
{
    PosType ret;
    if(di == 1)     //东面的通道块
    {
        ret.x = curpos.x + 1;
        ret.y = curpos.y;
    }else if(di == 2)   //南面的通道块
    {
        ret.x = curpos.x;
        ret.y = curpos.y + 1;

    }else if(di == 3)     //西面的通道块
    {
        ret.x = curpos.x - 1;
        ret.y = curpos.y;
    }else if(di ==4)       //北面的通道块
    {
        ret.x = curpos.x;
```

```
        ret.y = curpos.y - 1;
    }else
        assert(0);

    return ret;

}

// MazePath 计算从迷宫入口到出口的路径，其中参数：
// maze是个二维的迷宫矩阵，start是迷宫的出发点，end是迷宫的出口
bool MazePath(int* maze,PosType start,PosType end)
{
```

③

```
}
int main(int argc, char* argv[])
{
    //构造迷宫
    //0----没有走过,3----是墙壁
    int Maze[MAZEWIDTH][MAZEHEIGHT]=
    {
        3,3,3,3,3,3,3,3,3,3,
        3,0,0,3,0,0,0,3,0,3,
        3,0,0,3,0,0,0,3,0,3,
        3,0,0,0,0,3,3,0,0,3,
        3,0,3,3,3,0,0,0,0,3,
        3,0,0,0,3,0,0,0,0,3,
        3,0,3,0,0,0,3,0,0,3,
        3,0,3,3,3,0,3,3,0,3,
        3,3,0,0,0,0,0,0,0,3,
        3,3,3,3,3,3,3,3,3,3
    };
    //设置起始通道块和结束通道块
    PosType start,end;
    start.x = 1;
    start.y = 1;
    end.x = 8;
    end.y = 8;
    bool success =  MazePath((int*)Maze, start,end);
    getchar();
    return 0;
```

```
}
```

8.（13 分）简答以下有关 C++语言的问题：

（1） 比较类的三种继承方式 public（公有继承）、protected（保护继承）和 private（私有继承）之间的差别。

（2） 如果类 A 是类 B 的友元，类 B 是类 C 的友元，类 D 是类 A 的派生类，那么类 B 是类 A 的友元吗？类 C 是类 A 的友元吗？类 D 是类 B 的友元吗？简述理由。

（3） 什么叫多态性？C++支持多态的主要方式是什么？

9.（17 分）编写一个 C++程序，满足以下要求：

（1） 定义一个 Shape 基类，在此基础上派生出名为 Rectangle 的矩形类和名为 Circle 的圆形类，二者都有 GetArea()函数计算对象的面积。由 Rectangle 类派生名为 Square 的正方形类。

（2） Rectangle 类有宽度和长度属性（其初值分别为 2 和 3），Circle 类有半径属性。

（3） 在程序输出中：

（a） 直接输出矩形的面积。

（b） 输出宽度、长度分别为 4 和 5 的矩形的面积。

（c） 输出半径为 6 的圆形的面积。

（d） 输出边长为 7 的正方形的面积。

2004 年专业基础课参考答案

1.（10 分）

（1） 以 n、a_i(i=0,1, . . . , n)、x_0 作为输入，为了进行一元 n 次多项式 $P_n(x)=a_0x^n+a_1x^{n-1}+a_2x^{n-2}+\ldots+a_{n-1}x+a_n$ 在 x_0 点的值 $P_n(x_0)$的计算，请给出你认为效率最好的算法。

参考答案：

```
sum = a0;
for(int i=1;i<=n;i++)
{
   sum = sum * x0 + ai;
}
```

（2）给出上述算法的基本操作、基本操作执行次数和时间复杂度。

参考答案：

基本操作： sum = sum * x_0 + a_i

基本操作执行次数：n

时间复杂度：O(n)

2.（10 分）

设有三对角矩阵（a_{ij}）nxn，将其三条对角线上的元素逐行地存于数组 B[3n-2]中，使得 B[k]=a_{ij}，求：

（1）用 i，j 表示 k 的下标变换公式；

（2）用 k 表示 i，j 的下标变换公式。

参考答案：

k = 2* i + j – 3 (|i-j|≤1)

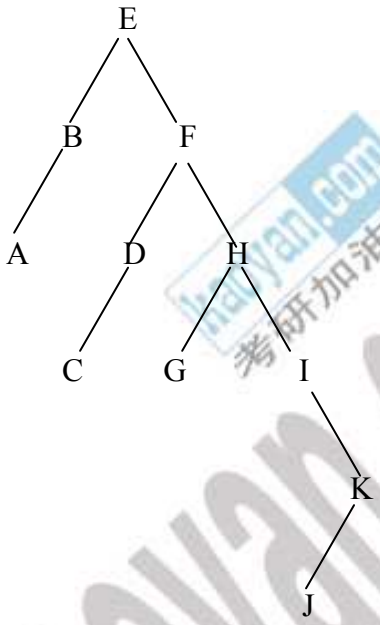
i = [(k+1)/3] + 1 （0≤k≤3n-1）

j = k+1 – 2 * [k/3]

3.（10 分）

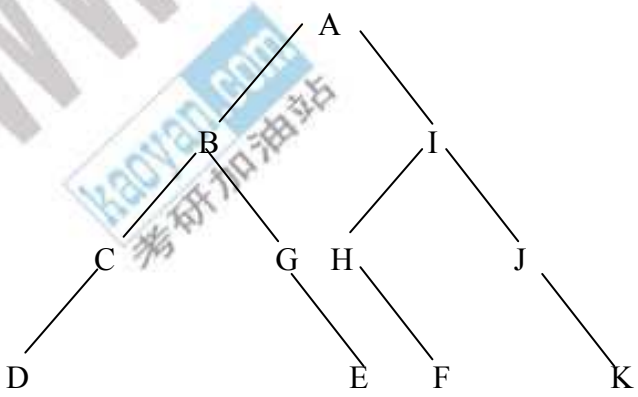
（1） 已知一棵二叉树的先序序列为 EBADCFHGIKJ 和中序序列为 ABCDEFGHIJK，请画出该树，并给出计算或推理过程。

参考答案：



（2） 已知一棵二叉树的中序序列为 DCBGEAHFIJK 和后序序列为 DCEGBFHKJIA，请画出该树，并给出计算或推理过程。

参考答案：



4．（15 分）

某人自下往上走完一个 N 级的台阶，每步只能走一级或两级台阶：

（1） 给出能够计算出上述台阶所有走法的递归算法。

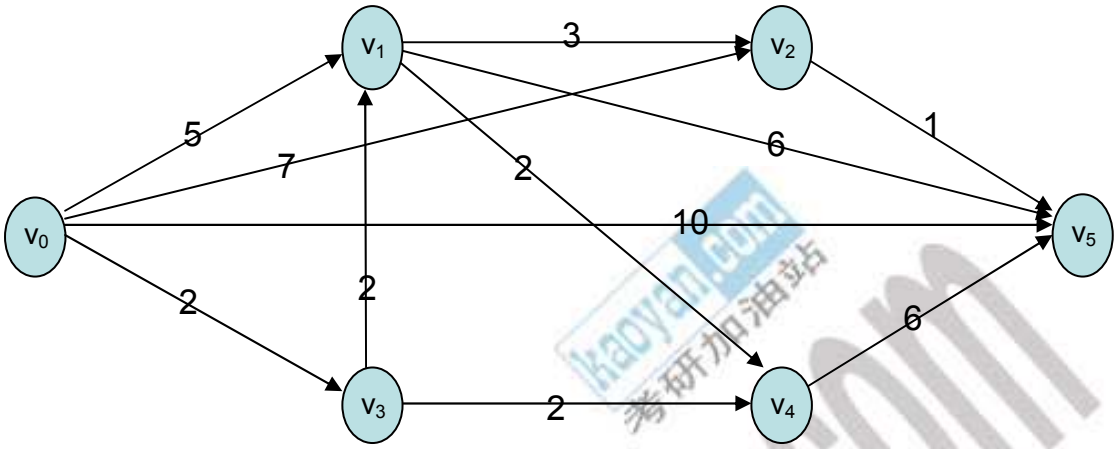
（2） 以 C 或 C^{++}实现上述算法。

参考答案：

$$
\text{walk}(n) = \begin{cases} \text{step}(1) & \text{如果 } n = 1 \\ \{\text{step}(1) \,\|\, \text{step}(1)\} \cup \text{step}(2) & \text{如果 } n = 2 \\ \{\text{step}(1) \,\|\, \text{walk}(n-1)\} \cup \{\text{step}(2) \,\|\, \text{walk}(n-2)\} & \text{如果 } n \geqslant 3 \end{cases}
$$

5．（20 分）

下图是一个有向图，其中每条弧段上的数字表示该弧段的权值。请用 Dijkstra 算法计算 v_0 到各点的最短路径及路径的长度（要求给出计算过程）。



参考答案：

v_0 到 v_1 的最短路径是（v_0,v_3,v_1）,最短路径的长度为 4

v_0 到 v_2 的最短路径是（v_0,v_2）或（v_0,v_3,v_1,v_2）,最短路径的长度为 7

v_0 到 v_3 的最短路径是（v_0,v_3）,最短路径的长度为 2

v_0 到 v_4 的最短路径是（v_0,v_3,v_4）,最短路径的长度为 4

v_0 到 v_5 的最短路径是（v_0,v_3,v_1,v_2,v_5）或（v_0,v_2, ,v_5）,最短路径的长度为 8

6.（30 分）

已知如下所示长度为 12 的表

（Jan，Feb，Mar，Apr，May，June，July，Aug，Sep，Oct，Nov，Dec）

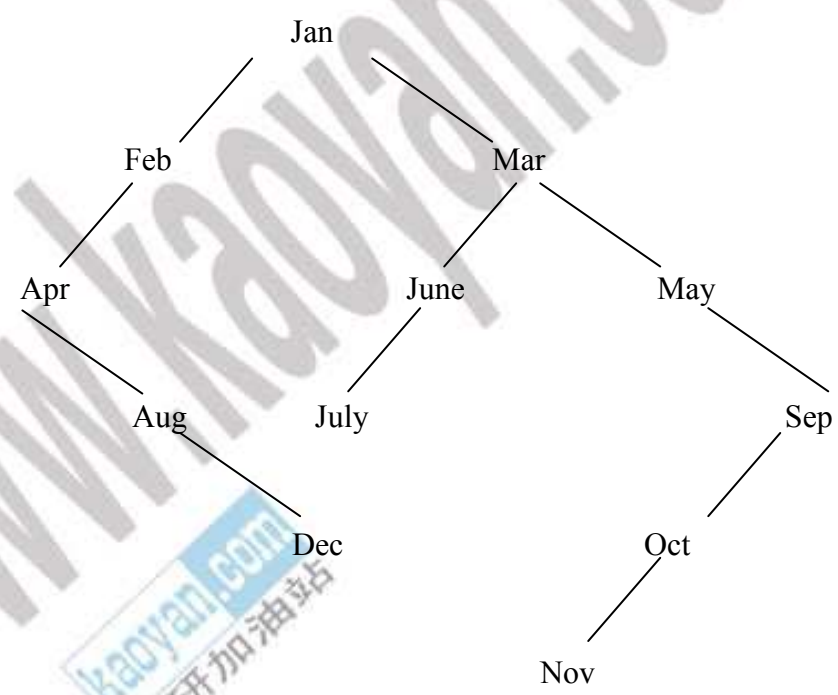
（1）试按表中元素的顺序依次插入一棵初始为空的二叉排序树，画出插入完成之后的二叉排序树，并求其在等概率情况下查找成功的平均查找长度。

（2）若对表中元素先进行排序构成有序表，求在等概率情况下对此有序表进行折半查找时查找成功的平均查找长度。

（3）请按表中元素的顺序构造一棵平衡二叉排序树，并求其在等概率情况下查找成功的平均查找长度。

参考答案：

（1）



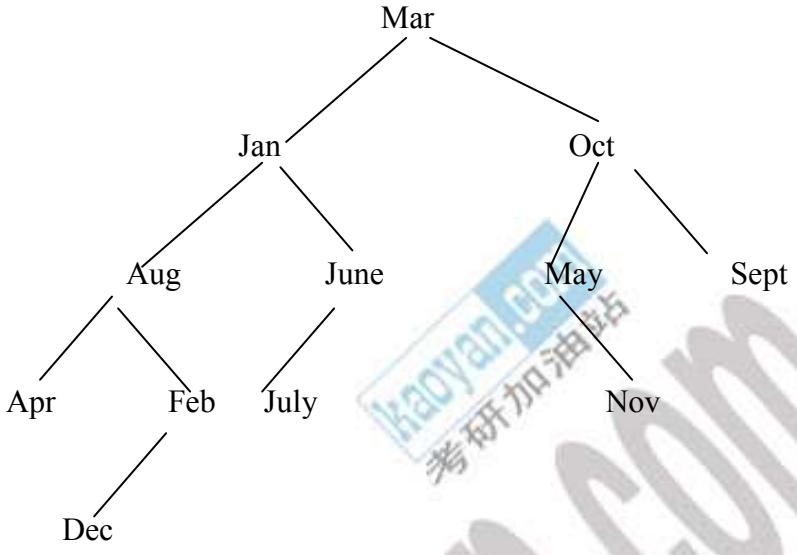
在等概率情况下平均查找长度=（1*1+2*2+3*3+4*3+5*2+6*1）/12 =7/2

（2）经排序后的表及在折半查询时找到表中元素所需比较的次数为：

Apr	Aug	Dec	Feb	Jan	July	June	Mar	May	Nov	Oct	Sept
3	4	2	3	4	1	3	4	2	4	3	4

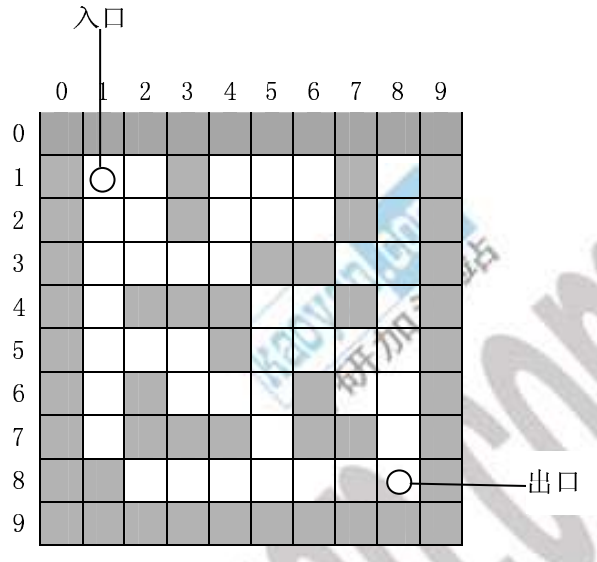
在等概率情况下平均查找长度=（1*1+2*2+3*4+4*5）=37/12

（3）



平衡二叉排序树平均查找长度 (1*1+2*2+3*4+4*3+5*1)/12= 19/6

7.（25 分）如图所示的方块图表表示一个迷宫。图中的每个白方块表示为通道，黑方块为墙。请在①、②、③处填充必要的 C 语言代码，完成下面求从迷宫入口到出口路径的程序。




```
#define STACK_INCREMENT 10                  //栈每次增加的大小
#define OK true
#define ERROR false
#define MAZEWIDTH 10                        //迷宫的X方向大小
#define MAZEHEIGHT 10                       //迷宫的Y方向大小

 //坐标位置状态,0----没有走过，1----走过了，2----不通,3----是墙壁
enum status {
    NOT_PASSED,   //没有走过该通道块
    PASSED,       //该通道块已经走过了
    NOT_THROUGH,  //不通
    IS_WALL       //是墙壁
};

typedef struct postype
{
    int x;  //横坐标
    int y;  //纵坐标
}PosType;

typedef struct selemtype
{
    int ord;          //通道块在路经中的"序号"
    PosType seat;     //通道块在迷宫中的"坐标位置"
```

```
    int di;          //从此通道块走向下一个通道块的"方向"
                     //1----东面，2---南面，3---西面，4---北面
}SElemType;    //栈的元素类型

typedef struct
{
    SElemType *base;   //栈底
    SElemType *top;     //栈顶
    int stacksize;     //栈大小
}SqStack;              //栈结构

//构造一个空栈
bool InitStack(SqStack &S)
{
    S.base = (SElemType *)malloc(STACK_INIT_SIZE * sizeof(SElemType));

    if(!S.base) return ERROR;
    S.top = S.base;
    S.stacksize = STACK_INIT_SIZE;
    return OK;
}

//判断栈是否为空
bool StackEmpty(SqStack S)
{
    if(S.base == S.top)
        return true;
    else
        return false;
}

//插入元素E为新的栈顶元素
bool Push(SqStack &S,SElemType e)
{
```

①

```
}
```

```
//若栈不空，则删除S的栈顶元素，用E返回其值，并返回OK，否则返回ERROR
bool Pop(SqStack &S,SElemType &e)
{
```

②

```
}

//能否通过curpos位置的通道块
bool Pass(int *maze,PosType curpos)
{
    if(maze[curpos.x * MAZEWIDTH + curpos.y] == NOT_PASSED)
        //当前的还没有走过
        return true;
    else
        return false;
}

//maze是个二维的迷宫矩阵
//curpos是当前的通道块
//di是下一个通道块的方向
PosType NextPos(int *maze,PosType curpos,int di)
{
    PosType ret;
    if(di == 1)      //东面的通道块
    {
        ret.x = curpos.x + 1;
        ret.y = curpos.y;
    }else if(di == 2)   //南面的通道块
    {
        ret.x = curpos.x;
        ret.y = curpos.y + 1;

    }else if(di == 3)      //西面的通道块
    {
        ret.x = curpos.x - 1;
        ret.y = curpos.y;
    }else if(di ==4)        //北面的通道块
    {
        ret.x = curpos.x;
        ret.y = curpos.y - 1;
```

```
    }else
        assert(0);

    return ret;

}

// MazePath 计算从迷宫入口到出口的路径，其中参数：
// maze是个二维的迷宫矩阵，start是迷宫的出发点，end是迷宫的出口
bool MazePath(int* maze,PosType start,PosType end)
{
```

③

```
}
int main(int argc, char* argv[])
{
    //构造迷宫
    //0----没有走过,3----是墙壁
    int Maze[MAZEWIDTH][MAZEHEIGHT]=
    {
        3,3,3,3,3,3,3,3,3,3,
        3,0,0,3,0,0,0,3,0,3,
        3,0,0,3,0,0,0,3,0,3,
        3,0,0,0,0,3,3,0,0,3,
        3,0,3,3,3,0,0,0,0,3,
        3,0,0,0,3,0,0,0,0,3,
        3,0,3,0,0,0,3,0,0,3,
        3,0,3,3,3,0,3,3,0,3,
        3,3,0,0,0,0,0,0,0,3,
        3,3,3,3,3,3,3,3,3,3
    };
    //设置起始通道块和结束通道块
    PosType start,end;
```

```
    start.x = 1;
    start.y = 1;
    end.x = 8;
    end.y = 8;
    bool success =  MazePath((int*)Maze, start,end);
    getchar();
    return 0;
}
```

参考答案

①

```
if(S.top - S.base >= S.stacksize){

        S.base = (SElemType *)realloc(S.base,
            (S.stacksize + STACK_INCREMENT)*sizeof(SElemType));
        if(!S.base) return ERROR;
        S.top = S.base + S.stacksize;
        S.stacksize += STACK_INCREMENT;
    }

    *S.top = e;
    S.top++;
    return OK;
```

②

```
    if(S.top == S.base) return ERROR;
    S.top --;
    e = *S.top;
    return OK;
```

③

```
    SqStack S;
    InitStack(S);
    PosType curpos = start;
    int curStep = 1;
    do{

        if(Pass(maze,curpos)){
            //如果curpos通道块可以通过(没有走过此通道块，也不是墙壁)
            //标记已经走过
```

```
            maze[curpos.x * MAZEWIDTH + curpos.y] = PASSED;
            SElemType e;
            e.ord = curStep;
            e.seat.x = curpos.x;
            e.seat.y = curpos.y;
            e.di = 1;
            Push(S,e);
            if(curpos.x == end.x && curpos.y == end.y){
            //到达迷宫出口
                int i = 0;
                while(!StackEmpty(S))
                {
                    i++;
                    SElemType e;
                    Pop(S,e);
                    printf("%d, %d %d\n",i,e.seat.x,e.seat.y);
                }
                return true;
            }
            //搜索东面的通道块
            curpos = NextPos(maze,curpos,1);
            curStep++;
        }//if pass
        else{
            //如果curpos通道块不可以通过（已经走过此通道块，或者是墙壁）
            if(!StackEmpty(S)){
                SElemType e;
                Pop(S,e);
                //从栈中推出不能通过的通道块
                while(e.di == 4 && !StackEmpty(S)) {
                    //标记不通
                    maze[e.seat.x * MAZEWIDTH + e.seat.y] = NOT_THROUGH;
                    Pop(S,e);
                }
                //换下一个方向搜索
                if(e.di < 4){
                    e.di++;
                    Push(S,e);
                    curStep = e.ord + 1;
                    curpos = NextPos(maze,e.seat,e.di);
                }
            }
        }
    }while(!StackEmpty(S));
```

```
return false;
```

8.（13 分）简答以下有关 C++语言的问题：

（1） 比较类的三种继承方式 public（公有继承）、protected（保护继承）和 private（私有继承）之间的差别。

参考答案：

公有继承：基类的 public（公有）和 protected（保护）成员的访问属性在派生类中不变，而基类的 private（私有）成员不可访问；

私有继承：基类的 public（公有）和 protected（保护）成员都以 private（私有）成员身份出现在派生类中，而基类的 private（私有）成员不可访问；

私有继承：基类的 public（公有）和 protected（保护）成员都以 protected（保护）成员的身份出现在派生类中，而基类的 private（私有）成员不可访问。

（2） 如果类 A 是类 B 的友元，类 B 是类 C 的友元，类 D 是类 A 的派生类，那么类 B 是类 A 的友元吗？类 C 是类 A 的友元吗？类 D 是类 B 的友元吗？简述理由。

参考答案：

类 B 不是类 A 的友元，友元关系不具有交换性；

类 C 不是类 A 的友元，友元关系不具有传递性；

类 D 不是类 B 的友元，友元关系不能被继承。

（3） 什么叫多态性？C++支持多态的主要方式是什么？

参考答案：

多态性是指同样的消息被不同类型的对象接收时导致完全不同的行为。

C++支持多态的主要方式是重载和虚函数。

9.（17 分）编写一个 C++程序，满足以下要求：

（2） 定义一个 Shape 基类，在此基础上派生出名为 Rectangle 的矩形类和名为 Circle 的圆形类，二者都有 GetArea()函数计算对象的面积。由 Rectangle 类派生名为 Square 的正方形类。

（2） Rectangle 类有宽度和长度属性（其初值分别为 2 和 3），Circle 类有半径属性。

（3） 在程序输出中：

（a） 直接输出矩形的面积。

（b） 输出宽度、长度分别为 4 和 5 的矩形的面积。

（c） 输出半径为 6 的圆形的面积。

（d） 输出边长为 7 的正方形的面积。

参考答案：

```
#include <iostream.h>
#define PI 3.14159

class Shape
{
public:
    Shape(){}
    ~Shape(){}
    virtual float GetArea(){return -1;}
};

class Circle:public Shape
{
public:
    Circle(float radius):itsRadius(radius){}
    ~Circle(){}
    float GetArea(){return (float)PI*itsRadius*itsRadius;}
private:
    float itsRadius;
};

class Rectangle:public Shape
{
public:
    Rectangle();
    Rectangle(float len,float width):itsLength(len),itsWidth(width){}
    ~Rectangle(){}
    float GetArea(){return itsLength*itsWidth;}
```

```
private:
    float itsWidth;
    float itsLength;
};

Rectangle::Rectangle()
{
    itsWidth=2;
    itsLength=3;
}

class Square:public Rectangle
{
public:
    Square(float len);
    ~Square(){}
};

Square::Square(float len):Rectangle(len,len)
{
}

void main()
{
    Shape *sp;
    sp=new Rectangle();
    cout<<"The area of the rectangle is "<<sp->GetArea()<<endl;
    delete sp;

    sp=new Rectangle(4,5);
    cout<<"The area of the rectangle is "<<sp->GetArea()<<endl;
    delete sp;

    sp=new Circle(6);
    cout<<"The area of the circle is "<<sp->GetArea()<<endl;
    delete sp;

    sp=new Square(7);
    cout<<"The area of the square is "<<sp->GetArea()<<endl;
    delete sp;
}
```