Q1:

有一种数据结构 B2=（D，R），其中

D={48，25，64，57，82，36，75}

R={r1，r2}

r1={<25，36>，<36，48>，<48，57>，<57，

64>，<64，75>，<75，82>}

r2={<48，25>，<48，64>，<64，57>，<64，82>，

<25，36>，<82，75>}

画出其逻辑结构表示，指出是什么类型?

$A =$

	25	36	48	57	64	75	82
25	0	1	1	0	0	0	0
36	1	0	1	0	0	0	0
48	0	0	0	1	0	0	0
57	0	0	0	0	1	0	0
64	0	0	1	0	0	1	0
75	0	0	0	0	0	0	1
82	0	0	0	0	1	0	0

其中1表示两个节点之间存在关系，0表示不存在关系

结构类型应为图形结构

Q2：考虑下列两段描述，这两段描述均不能满足算法的特性，试问它们违反了哪些特性?

（1） 描述 1

```
void exam1()
{   int   n = 2;
   while (n%2 = = 0) n
           = n+2;
    printf("%d\n"，n);
}
```

变量n的值在函数中始终为偶数，该函数不会停止，违反了有穷性

（2） void exam2()

```
    {   int x，y;
        y=0;
       x=5/y; printf("%d，%d\n"，x，
  y);
}
```

在上述函数中x=5/y中y被赋值为0，未执行基本的运算规则，违反了可行性

Q3：某算法的时间复杂度为 O(n2)，表明该算法的（C ）。

A.问题规模是 n2　　B.执行时间等于 n2

C.执行时间与 n2 成正比　　D.问题规模与 n2 成正比

FDS-BCS 石雨宸 D23090120503
BCS009 Assignment 1

Q4: 下面几种算法时间复杂度中，时间复杂度最高的是（B）。
A.O(nlog$_2$n)　B.O(n^2)　C.O(n)　D.O(2n)

Q7: 算法的空间复杂度是指（D）。
A.算法中输入数据所占用的存储空间的大小
B.算法本身所占用的存储空间的大小
C.算法中所占用的所有存储空间的大小
D.算法中需要的辅助变量所占用存储空间的大小

Q5: 某算法的空间复杂度为 O(1)，则（ B ）。
A.该算法执行不需要任何辅助空间
B.该算法执行所需辅助空间大小与问题规模 n 无关
C.该算法执行不需要任何空间
D.该算法执行所需空间大小与问题规模 n 无关

Q6： 设计一个算法：求一元二次方程$ax^2 + bx + c$的根。

```c
#include <stdio.h>
#include <math.h>

// 复数结构体
typedef struct
{
    double real;
    double imaginary;
} Complex;

// 计算一元二次方程的根
void quadratic_roots(double a, double b, double c,
Complex *root1, Complex *root2)
{
    // 计算判别式
    double d = b * b - 4 * a * c;
    // 产生两个不相等的实根
    if (d > 0)
    {
        root1->real = (-b + sqrt(d)) / (2 * a);
        root1->imaginary = 0;
        root2->real = (-b - sqrt(d)) / (2 * a);
        root2->imaginary = 0;
```

```
    }
    // 产生两个相等的实根
    else if (d == 0)
    {
        root1->real = -b / (2 * a);
        root1->imaginary = 0;
        root2->real = -b / (2 * a);
        root2->imaginary = 0;
    }
    // 产生两个共轭复根
    else
    {
        root1->real = -b / (2 * a);
        root1->imaginary = sqrt(-d) / (2 * a);
        root2->real = -b / (2 * a);
        root2->imaginary = -sqrt(-d) / (2 * a);
    }
}
// 打印复数
void print(const char *prefix, Complex c)
{
    if (c.imaginary == 0)
    {
        printf("%s %f\n", prefix, c.real);
    }
    else if (c.imaginary > 0)
    {
        printf("%s %f + %fi\n", prefix, c.real,
c.imaginary);
    }
    else
    {
        printf("%s %f - %fi\n", prefix, c.real, -
c.imaginary);
    }
}

int main()
{
    // 输入系数
    double a, b, c;
```

```
    printf("Enter a, b, c:");
    scanf("%lf %lf %lf", &a, &b, &c);
    Complex root1, root2;
    quadratic_roots(a, b, c, &root1, &root2);
    print("Root1:", root1);
    print("Root2:", root2);
    return 0;
}
```

Q7： 设计一个算法求 1+(1+2)+(1+2+3)+…+(1+2+3+…n), n>2。

```
#include <stdio.h>
#include <math.h>
// 求和
int sum(int n)
{
    int totalSum = 0;
    for (int i = 1; i <= n; i++)
    {
        int currSum = (i * (i + 1)) / 2;
        totalSum += currSum;
    }
    return totalSum;
}
int main()
{
    int n;
    printf("Enter n:");
    scanf("%d", &n);
    if (n <= 2)
    {
        printf("Error\n");
        return 1;
    }

    int result = sum(n);
    printf("Result: %d\n", result);

    return 0;
}
```