浙江理工大学 2017—2018 学年第 2 学期

《C#程序设计》期末试卷（A）卷标准答案和评分标准

一、单选题（本大题共 32 分，每小题 2 分）

1	2	3	4	5
D	C	C	B	C
6	7	8	9	10
C	A	B	A	C
11	12	13	14	15
A	A	C	C	A
16				
B				

二、程序设计题（共 68 分）

1、

(1)

```
class Employee
{
    public string Name { get; set; }
    public DateTime Birthday { get; set; }
    public double Salary { get; set; }
    public override string ToString()
    {
        return string.Format("{0}: Salary={1}, Birthday={2}", Name, Salary, Birthday);
    }
}
```

(2)

```
    private static List<Employee> FindEmployee(Employee[] ps, EmployeePredicate t)
    {
        List<Employee> results = new List<Employee>();
        foreach (Employee p in ps)
        {
            if (t(p))
                results.Add(p);
        }
        return results;
    }
```

(3)

```
    private static bool FilterBySalary(Employee emp)
    {
        return emp.Salary >= 7000;
    }
```

(4)

```
            List<Employee> high = FindEmployee(employees, FilterBySalary);
            foreach(Employee emp in high)
                Console.WriteLine(emp);
```

(5)

```
    public static class MyExtensions
    {
        public static bool IsEighties(this DateTime d)
        {
            return d.Year > 1979 && d.Year < 1980;
        }
    }
```

(6)

```
            List<Employee> eighties = FindEmployee(employees, p => p.Birthday.IsEighties());
            foreach(Employee emp in eighties)
                Console.WriteLine(emp);
```

(7)

```
            var eighties2 = from emp in employees
                            where emp.Birthday.IsEighties() == true
                            orderby emp.Salary descending
                            select emp;
            foreach (Employee emp in eighties2)
                Console.WriteLine(emp);
```

2、

```
    class Employee : IComparable<Employee>
    {
        public string Name { get; set; }
        public DateTime Birthday { get; set; }
        public double Salary { get; set; }
        public override string ToString()
        {
            return string.Format("{0}: Salary={1}, Birthday={2}", Name, Salary, Birthday);
        }

        public int CompareTo(Employee emp)
        {
            return (int)(Salary - emp.Salary);
        }
    }

    class EmployeeComparer : IComparer<Employee>
    {
        public int Sort { get; set; }
        public EmployeeComparer(int sort)
        {
            Sort = sort;
        }
        public int Compare(Employee e1, Employee e2)
        {
            switch(Sort)
            {
                case 0:
                    return (int)(e1.Salary - e1.Salary);
                case 1:
                    return e1.Name.CompareTo(e2.Name);
```

```
            default:
                return e1.Birthday.CompareTo(e2.Birthday);
        }
    }
}
```

3

```
enum Position
{
    Inside, On, Outside
}
class Circle
{
    public int X { get; set; }
    public int Y { get; set; }
    public int Radius { get; set; }
    public Position Contains(int x, int y)
    {
        double dist = Math.Sqrt((X - x) * (X - x) + (Y - y) * (Y - y));
        if (dist < Radius)
            return Position.Inside;
        else if (dist == Radius)
            return Position.On;
        else
            return Position.Outside;
    }
    public List<Circle> Mirrors(int axis)//0
    {
        List<Circle> circles = new List<Circle>();
        switch (axis)
        {
            case 0:
                circles.Add(Mirror(true));
                circles.Add(Mirror(false));
                circles.Add(Mirror(true).Mirror(false));
                break;
            case 1:
                circles.Add(Mirror(true));
                break;
            case 2:
                circles.Add(Mirror(false));
                break;
        }
        return circles;
    }

    public Circle Mirror(bool xAxis)
    {
        if (xAxis)
            return new Circle { X = X, Y = -Y, Radius = Radius };
        else
            return new Circle { X = -X, Y = Y, Radius = Radius };
    }
}
```