1. https://www.hackerrank.com/challenges/plus-minus/problem?isFullScreen=true

```
class Result
{

    /*
     * Complete the 'plusMinus' function below.
     *
     * The function accepts INTEGER_ARRAY arr as parameter.
     */

    public static void plusMinus(List<int> arr)
    {
        int Pos = 0, Neg=0, Zero=0;
        int Len = arr.Count();
        foreach(int i in arr)
        {
            if(i==0)
            {
                Zero++;
            }
            else if(i>0)
            {
                // Console.WriteLine(i);
                Pos++;
            }
            else
            {
                Neg++;
            }
        }
        // float ans = ((float)Pos / Len);
        Console.WriteLine(((float)Pos / Len).ToString("F6"));
        Console.WriteLine(((float)Neg / Len).ToString("F6"));
        Console.WriteLine(((float)Zero / Len).ToString("F6"));
        // Console.WriteLine(Len);
    }
```

```
}
```

2. https://www.hackerrank.com/challenges/staircase/problem?isFullScreen=true

```
class Result
{

    /*
     * Complete the 'staircase' function below.
     *
     * The function accepts INTEGER n as parameter.
     */

    public static void staircase(int n)
    {
        for(int i=1;i<=n;i++)
        {
            for(int j=i;j<n;j++)
            {
                Console.Write(" ");
            }
            for(int k=0;k<i;k++)
            {
                Console.Write("#");
            }
            Console.Write("\n");
        }
    }

}
```

3. https://www.hackerrank.com/challenges/mini-max-sum/problem?isFullScreen=true

```
class Result
{

    /*
     * Complete the 'miniMaxSum' function below.
     *
```

```
     * The function accepts INTEGER_ARRAY arr as parameter.
     */

    public static void miniMaxSum(List<int> arr)
    {
        List<long>longArr = arr.Select(x=> (long)x).ToList();

        longArr.Sort();
        long sum = longArr.Sum(); // Now it's a long sum
        long max = sum - longArr[0]; // Exclude the smallest value
        long min = sum - longArr[4]; // Exclude the largest value

        Console.WriteLine(min + " " + max);
    }

}
```

4. https://www.hackerrank.com/challenges/birthday-cake-candles/problem?isFullScreen=true

```
class Result
{

    /*
     * Complete the 'birthdayCakeCandles' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts INTEGER_ARRAY candles as parameter.
     */

    public static int birthdayCakeCandles(List<int> candles)
    {
        candles.Sort();
        int n = candles.Count();
        List<int>tall = candles.Where(x => x ==
candles[n-1]).ToList();

        return tall.Count();
    }
```

```
}
```

5. https://www.hackerrank.com/challenges/time-conversion/problem?isFullScreen=true

```
class Result
{

    /*
     * Complete the 'timeConversion' function below.
     *
     * The function is expected to return a STRING.
     * The function accepts STRING s as parameter.
     */

    public static string timeConversion(string s)
    {
        string ampm = s.Substring(8,2);
        string hours = s.Substring(0,2);
        int h = int.Parse(hours);
        if(ampm=="AM")
        {
            if(h==12)
            {
                hours="00";
            }
        }
        else
        {
            if(h!=12)
            {
                h = h + 12;
                hours = h.ToString("D2");
            }
        }

        string result = hours + s.Substring(2, 6);

        return result;
```

```
    }

}
```

6. https://www.hackerrank.com/challenges/grading/problem?isFullScreen=true

```
class Result
{

    /*
     * Complete the 'gradingStudents' function below.
     *
     * The function is expected to return an INTEGER_ARRAY.
     * The function accepts INTEGER_ARRAY grades as parameter.
     */

    public static List<int> gradingStudents(List<int> grades)
    {
        List<int> ans = new List<int>();

        foreach(int i in grades)
        {
            if(i>=38)
            {
                int rem = i%5;
                if(rem>=3)
                {
                    int quo = i/5;
                    ans.Add((quo+1)*5);
                }
                else
                {
                    ans.Add(i);
                }
            }
            else
            {
                ans.Add(i);
            }
```

```
        }
        return ans;
    }

}
```

7. https://www.hackerrank.com/challenges/apple-and-orange/problem?isFullScreen=true

```
class Result
{

    /*
     * Complete the 'countApplesAndOranges' function below.
     *
     * The function accepts following parameters:
     *  1. INTEGER s
     *  2. INTEGER t
     *  3. INTEGER a
     *  4. INTEGER b
     *  5. INTEGER_ARRAY apples
     *  6. INTEGER_ARRAY oranges
     */

    public static void countApplesAndOranges(int s, int t, int a, int
b, List<int> apples, List<int> oranges)
    {
        int count=0;
        foreach(int i in apples)
        {
            if(a+i>=s && a+i<=t)
            {
                count++;
            }
        }
        Console.WriteLine(count);
        count=0;
        foreach(int i in oranges)
        {
            if(b+i>=s && b+i<=t)
```

```
            {
                count++;
            }
        }
        Console.WriteLine(count);
    }

}
```

8. https://www.hackerrank.com/challenges/kangaroo/problem?isFullScreen=true

```
class Result
{

    /*
     * Complete the 'kangaroo' function below.
     *
     * The function is expected to return a STRING.
     * The function accepts following parameters:
     *  1. INTEGER x1
     *  2. INTEGER v1
     *  3. INTEGER x2
     *  4. INTEGER v2
     */

    public static string kangaroo(int x1, int v1, int x2, int v2)
    {
        if (v1 == v2)
    {
        return x1 == x2 ? "YES" : "NO";
    }

    int diffPosition = x2 - x1;
    int diffVelocity = v1 - v2;

    // Check if they meet after a number of jumps
    if (diffVelocity != 0 && (diffPosition % diffVelocity == 0) &&
(diffPosition / diffVelocity) >= 0)
    {
```

```
            return "YES";
        }

        return "NO";

        }

}
```

9. https://www.hackerrank.com/challenges/between-two-sets/problem?isFullScreen=true

```
class Result
{

    /*
     * Complete the 'getTotalX' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts following parameters:
     *  1. INTEGER_ARRAY a
     *  2. INTEGER_ARRAY b
     */

    public static int getTotalX(List<int> a, List<int> b)
    {
        List<int>ans = new List<int>();
        List<int>ans1 = new List<int>();

        for(int i=a[a.Count()-1];i<=b[0];i++)
        {
            int count=0;
            foreach(int j in a)
            {
                if(i%j==0)
                {
                    count++;
                }
            }
            if(count==a.Count())
```

```
            {
                ans.Add(i);
            }
        }

        foreach(int i in ans)
        {
            int count=0;
            foreach(int j in b)
            {
                if(j%i==0)
                {
                    count++;
                }
            }
            if(count==b.Count())
            {
                ans1.Add(i);
            }
        }

        return ans1.Count();

    }

}
```

10. https://www.hackerrank.com/challenges/breaking-best-and-worst-records/problem?isFullScreen=true

```
class Result
{

    /*
     * Complete the 'breakingRecords' function below.
     *
     * The function is expected to return an INTEGER_ARRAY.
     * The function accepts INTEGER_ARRAY scores as parameter.
     */
```

```
    public static List<int> breakingRecords(List<int> scores)
    {
        List<int>min_max = new List<int>();
        List<int>ans1 = new List<int>();
        min_max.Add(scores[0]);
        min_max.Add(scores[0]);
        ans1.Add(0);
        ans1.Add(0);

        for(int i=1;i<scores.Count();i++)
        {
            if(scores[i]>min_max[1])
            {
                min_max[1]=scores[i];
                ans1[0] = ans1[0]+1;
            }
            if(scores[i]<min_max[0])
            {
                min_max[0]=scores[i];
                ans1[1] = ans1[1]+1;
            }
        }

        return ans1;
    }

}
```