

問題 1 以下の A01 は 1 から 12367 までの整数の逆数の和を求めるプログラムである。A02 は 1 から n までの逆数の和が 10.0 以上になる n の最小値を求めるプログラムである。空欄を適切に埋めよ。(1)には変数 sum を宣言し初期化する式を、(2)にはループを制御する変数 i の宣言、継続条件、ステップを進める式などを入れること。(3)にはループ継続の条件を入れること。

```
public class A01 {
    public static void main(String[] args){
        (1) ;
        for( (2) )
            sum += 1.0 / i;

        System.out.println("1/1 + 1/2 + ... + 1/12367 = " + sum);
    }
}
```

A01 の実行結果

```
1/1 + 1/2 + ... + 1/12367 = 10.000043008275778
```

```
public class A02 {
    public static void main(String[] args) {
        (1) ;
        int n=0;
        while( (3) ){
            n++;
            sum += 1.0/n;
        }
        System.out.println(
            "1 から n までの整数の逆数の和が 10.0 以上になるのは n >= "
            + n + " のときである。");
    }
}
```

A02 の実行結果

```
1 から n までの整数の逆数の和が 10.0 以上になるのは n >= 12367 のときである。
```

問題 2 プログラム A03 は普通郵便の重量を整数[g]で入力し、料金を出力する。main メソッドは入力に責任をもち、printRate メソッドは重量(整数 g)を引数にとり、料金あるいは警告のメッセージを表示する。右の表に従いプログラム中の空欄 (1)~(5)を適切に埋めよ。

重量[g]	料金[円]
25 以下	80
25 を越えて 50 以下	90
50 を超える	定形外扱い

```

public class A03 {
    public static void main(String[] args) throws IOException {
        BufferedReader reader
            = new BufferedReader(new InputStreamReader(System.in));

        System.out.print("重量[g]を整数で入力: ");
        int input = Integer.parseInt(reader.readLine());
        (1);
    }

    public static (2) printRate( (3) ){
        if(weight < 0)
            System.out.println("0 以上の整数を入力してください。");
            (4)
            System.out.println("料金は 80 円です。");
            (5)
            System.out.println("料金は 90 円です。");
        else
            System.out.println("定形外になります。");
    }
}

```

実行例 1

```

重量[g]を整数で入力: 100
定形外になります。

```

実行例 2

```

重量[g]を整数で入力: 20
料金は 80 円です。

```

問題 2(続き) プログラム A04 はユーザからみると A03 と同じ実行結果を与える。A04 においては main メソッドは重量(int 型)を入力をした後、getRate メソッドを呼び出し料金あるいは警告コード(int 型)を受け取る。main メソッドはその結果に基づいて料金あるいは警告を出力する。getRate メソッドは重量(整数[g])を引数にとり、料金あるいは警告コード(いずれも整数)を返す。以下のプログラム中の空欄(6), (7)を適切に埋めよ。

```
public class A04 {
    public static void main(String[] args) throws IOException {
        BufferedReader reader
            = new BufferedReader(new InputStreamReader(System.in));

        System.out.print("重量[g]を整数で入力: ");
        int input = Integer.parseInt(reader.readLine());
        (6) ;

        if(rate == -1)
            System.out.println("0 以上の整数を入力してください。");
        else if(rate == -2)
            System.out.println("定形外になります。");
        else
            System.out.println("料金は" + rate + "円です。");
    }

    public static (7) getRate( (3) ){
        if(weight < 0)
            return -1;
        (4)
            return 80;
        (5)
            return 90;
        else
            return -2;
    }
}
```

問題 3 プログラム A05 は配列 x に格納された各整数データの偏差、つまりデータから平均値を引いたもの、を配列 y に格納し、小数点以下 2 桁で出力する。printf メソッドはデータを整形して出力するためのメソッドで、授業では扱わなかったが実行結果をみて、雰囲気が分かれば十分である。プログラム中の空欄(1)～(6)を適切に埋めよ。

```
public class A05 {
    public static void main(String[] args) {
        // int型配列xを宣言し、初期化
        (1) = {100, 70, 50, 45, 80, 60, 20};

        // 長さx.lengthのdouble型配列yを宣言する。
        (2) = new double[ (3) ];

        // 配列xのデータの平均を計算する
        int sum = 0;
        for( (4) )
            sum += (5) ;
        double average = (double) sum / x.length;

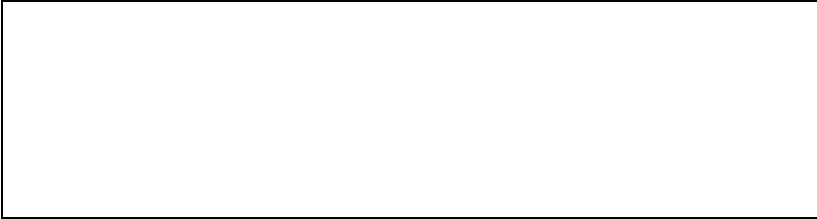
        // 配列yにxの各データの偏差を格納
        for(int i=0; i<x.length; i++)
            (6) = x[i] - average;

        for(int i=0; i<y.length; i++)
            System.out.printf("%6.2f ", y[i]);
    }
}
```

実行結果

```
39.29  9.29 -10.71 -15.71  19.29  -0.71 -40.71
```

問題 4 プログラム A06 は 5 および 6 の階乗を計算して結果を表示するプログラムである。factorial メソッドは整数を引数にとり、階乗の計算を実行し、その結果を整数で返す。引数は 0 以上であること、計算結果が int 型の範囲を超えないことを仮定してよい。

```
public class A06 {  
  
    public static void main(String[] args) throws IOException {  
        System.out.println("factorial(5) = " + factorial(5));  
        System.out.println("factorial(6) = " + factorial(6));  
    }  
  
    public static int factorial(int n){  
          
    }  
}
```

(1) プログラム内の空欄を埋めよ。以下の擬似コードに従って再帰呼び出しによって要求された処理を実現すること。

```
n = 0 なら 1 を返す  
そうでなければ n と factorial(n-1) の結果を掛けたものをかえす
```

(2) プログラム内の空欄を埋めよ。再帰呼び出しを使わずに for 文を用いて要求された処理を実現すること。

問題 5 以下のプログラム B01 の空欄を適切に埋めよ。

```
public class B01 {
    public static void main(String[] args) throws IOException {
        BufferedReader reader =
            new BufferedReader(new InputStreamReader(System.in));

        System.out.print("元本(円)は: ");
        double amount = Double.parseDouble(reader.readLine());

        System.out.print("利率(%)は: ");
        double rate = Double.parseDouble(reader.readLine());

        System.out.print(
            "1. 預ける年数を入力 2. 目標の金額を入力: ");
        int select = Integer.parseInt(reader.readLine());

        switch (select) {
            case 1:
                System.out.print("何年預けますか: ");
                int years = Integer.parseInt(reader.readLine());
                System.out.println(years + "年後は"
                    +  + "円になります。");
                break;
            case 2:
                System.out.print("目標の金額は: ");
                double target = Double.parseDouble(reader.readLine());
                System.out.println(
                     + "年後になります。");
                break;
            default:
                System.out.println("終了します。");
                break;
        }
    }
}
```

```
public static (3) getAmount ( (4) ) {
    double amount = init;
    for(int i=0; i<years; i++)
        (5) ;
    return amount;
}

public static (6) getYears ( (7) ) {
    int years = 0;
    double amount = init;
    while(amount < target){
        (5) ;
        years++;
    }
    return years;
}
}
```

getAmount メソッドは預金額の初期値(double 型、円単位で)、年利率(double 型、%で)、預ける年数 (int 型)を引数にとり、1年毎の複利で計算した指定年数経過後の元利合計 (double 型、円単位で)を返す。

getYears メソッドは預金額の初期値(double 型、円単位で)、目標とする元利合計(double 型、円単位で)、年利率(double 型、%)を引数にとり、1年毎の複利で計算した必要年数(int 型)を返す。

実行例を2つあげる。

```
元本 (円) は: 10000
利率 (%) は: 2.0
1. 預ける年数を入力 2. 目標の金額を入力: 1
何年預けますか: 10
10年後は 12189.944199947571 円になります。
```

```
元本 (円) は: 10000
利率 (%) は: 2.0
1. 預ける年数を入力 2. 目標の金額を入力: 2
目標の金額は: 20000
36年後になります。
```

問題 6 プログラム B02 は main メソッド内で定義された int 型配列 data において値が平均+10 以上のデータの個数を出力する。countGE メソッドは int 型配列と double 型実数を引数にとり、第 1 引数の配列における第 2 引数以上の要素の個数(int 型)を返す。sum メソッドは int 型配列を引数にとり、合計(int 型)を返す。プログラム中の空欄を適切に埋めよ。実行結果は以下の通り。

平均+10以上のデータの個数は 4

```
public class B02 {
    public static void main(String[] args) {
        int[] data = { 0, 10, 20, 30, 40, 50, 60, 70, 80, 90 };
        System.out.println("平均+10 以上のデータの個数は "
            + (1) );
    }

    public static (2) countGE( (3) ) {
        int count = 0;
        for (int i = 0; i < x.length; i++)
            if (x[i] >= y) count++;
        return count;
    }

    public static double average(int[] x) {
        return (double) sum(x) / (4);
    }

    public static (5) sum(int[] x) {
        (6)
        return total;
    }
}
```


問題 7 プログラム B03 は main メソッド内で定義された点(平面上)の配列 p の中から相互の距離が最大の点の組とその距離を求めるものである。sqDistance メソッドは 2 つの点 (いずれも double[] 型)を引数にとり、2 点間の距離の平方(double 型)を返す。実行結果は以下の通りで、まず最大距離が出力され、:] の後に 2 点のインデックスが昇順に出力される。最大距離を実現する 2 点の組が複数ある場合は最初に見つかったものが出力される。プログラム中の空欄を適切に埋めよ。

```
68.11754546370561:6 8
```

```
public class B03 {
    public static void main(String[] args) {
        double[][] p = { { 2, 8 }, { 4, 16 }, { 8, 31 },
            { 16, 3 }, { 31, 5 }, { 3, 23 }, { 5, 57 }, { 23, 11 },
            { 57, 13 }, { 11, 2 }, { 13, 4 } };

        int[] indexOfMaximum = new int[2];
        double maximum = 0;

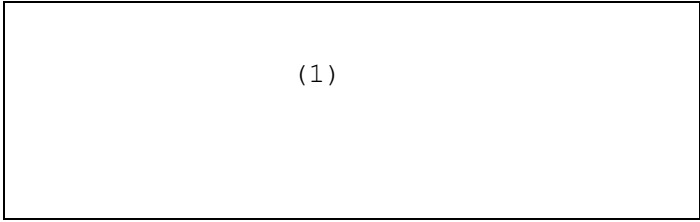
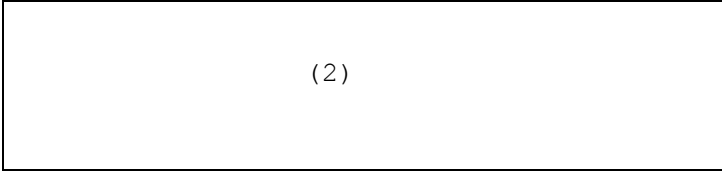
        for (int i = 0; i < p.length; i++) {
            for (int j = i + 1; j < p.length - 1; j++) {
                double d =  ;
                if (maximum < d) {
                    maximum =  ;
                    indexOfMaximum[0] =  ;
                    indexOfMaximum[1] =  ;
                }
            }
        }

        System.out.print(Math.sqrt(maximum) + ":");
        for (int i = 0; i < indexOfMaximum.length; i++) {
            System.out.print(indexOfMaximum[i] + " ");
        }

        System.out.println();
    }
}
//次ページに続く
```

```
public static (5) sqDistance( (6) ) {
    double d = (s[0] - t[0]) * (s[0] - t[0])
        + (s[1] - t[1]) * (s[1] - t[1]);
    return d;
}
}
```

問題 8 プログラム B04 は main メソッド内で定義された整数型配列 `sample` を降順にソートして出力する。`findMax` メソッドは走査開始インデックス `idx` と `int` 型配列 `a1` を引数にとり、`a1` の `idx` 番目以降の要素の中で最大のもののインデックスを返す。ただし、最大値が 2 個以上ある場合は小さいほうのインデックスを返す。`sort` メソッドは配列 `a` を引数にとり、降順ソートを実現する。プログラム中の空欄(1),(2)を次ページの擬似コードに従って適切に埋めよ。

```
public class B04 {  
    public static void main(String[] args) {  
        int[] sample = { 2, 4, 8, 16, 31, 3, 5, 23, 57, 11, 13 };  
        sort(sample);  
        for (int i = 0; i < sample.length; i++) {  
            System.out.print(sample[i] + " ");  
        }  
        System.out.println();  
    }  
  
    public static void sort(int[] a) {  
        for (int i = 0; i < a.length; i++) {  
              
            (1)  
        }  
    }  
  
    public static int findMax (int idx, int[] a1) {  
        int tmp = idx;  
          
        (2)  
        return tmp;  
    }  
}
```

```
sort(a)
```

```
  i = 0 から a の最後のインデックス-1 まで繰り返す  
    a の i 番目の要素と findMax(i) 番目の要素を交換する
```

```
findMax(idx, a1)
```

```
  tmp に idx を代入  
  i = idx + 1 から a1 の最後のインデックスまで繰り返す  
    if a[i] > a[tmp]  
      tmp に i を代入  
  tmp を返す
```

注意： 問題文および擬似コード中で「番目」という表現は、Java 言語の配列のインデックスにそのまま対応するものとする。

プログラム B04 の実行結果

```
57 31 23 16 13 11 8 5 4 3 2
```