学科名　　科　学籍番号　　氏名

問１ **次のプログラムの説明及びプログラムを読んで，［　　］に入れる適切な字句を，解答群の中から選べ。**

＜プログラムの説明＞

あらかじめプログラム中に用意した複数の整数の中から，コマンドラインで指定した整数を検索し，結果を表示するプログラムである。

1. コマンドラインから検索する整数（以下，検索値という）を指定して実行する。このとき，必ず一つの整数が指定されるものとする。また，コマンドラインで指定した値が整数でない場合は，考慮しないものとする。
2. 検索対象の整数はあらかじめ昇順に並べ，配列変数で用意する。
3. 検索は二分探索法で行い，配列中に検索値を発見した場合は検索値とその添字を表示する。発見できなかった場合もメッセージを表示する。
4. 二分探索法による探索の手順は，以下のとおりである。

①. 検索対象範囲を定める。最初は配列変数の全体が検索対象範囲である。

②. 検索対象範囲の中央の位置を求める。

③. 検索値と中央に位置する値とを比較する。

(a). 等しければ検索値の発見として検索を終了する。
(b). 検索値が中央の値より小さければ，中央の位置より小さい範囲を新しい検索対象範囲とする。
(c). 検索値が中央の値より大きければ，中央の位置より大きい範囲を新しい検索対象範囲とする。

④. 検索値を発見するか検索対象範囲がなくなるまで，②～③を繰り返す。

≪実行例≫

```
>java Q5 0
0 not found...
>java Q5 5
found 5 at index 2
>java Q5 10
found 10 at index 5
>java Q5 21
21 not found...
```

＜プログラム＞

```
1  public class Q5 {
2      public static void main(String[] args) {
3          int[] values = { 1, 3, 5, 6, 8, 10, 14, 20 };
4          int n = Integer.parseInt( [ (28) ] );
5
6          int low = 0, high = values.length - 1;
7          int mid;
8          do {
9              mid = [ (29) ];
10             if (n == values[mid]) {
11                 [ (30) ];
12             } else if (n < values[mid]) {
13                 [ (31) ];
14             } else {
15                 low = mid + 1;
16             }
17         } while (low <= high);
18
19         if ( [ (32) ] ) {
20             System.out.println("found " + values[mid]
21                     + " at index " + mid);
22         } else {
23             System.out.println(n + " not found...");
24         }
25     }
26 }
```

(28) の解答群

ア　args
イ　args[0]
ウ　args[1]
エ　args.first

(29) の解答群

ア　high / 2
イ　low / 2
ウ　(low - high) / 2
エ　(low + high) / 2

(30) の解答群

ア　high = mid
イ　low = mid
ウ　continue
エ　break

(31) の解答群

ア　low = mid
イ　low = mid - 1
ウ　high = mid
エ　high = mid - 1

(32) の解答群

ア　n == values[low]
イ　n == values[mid]
ウ　n == values[high]
エ　low > high

(28)	(29)	(30)	(31)	(32)
イ	エ	エ	エ	イ

学科名　　科　学籍番号　　氏名

問２ **次のプログラムの説明及びプログラムを読んで，［　　］に入れる適切な字句を，解答群の中から選べ。**

＜プログラムの説明＞

ある競技において，複数のプレイヤのデータを基にポイントの降順に並べ，順位，名前と共に表示するプログラムである。

1. 全プレイヤのデータは，名前，ポイントがあらかじめそれぞれプログラム内の配列変数に設定されている。

2. 順位は，自分より上位のプレイヤの人数＋１となるように順位付けをする。

3. このプログラムでは Q6 クラス，PlayerRanking クラス，Player クラスを定義する。

各クラスの役割は，次のとおりである。

Q6 クラス　　：実行用クラス
PlayerRanking クラス　：全プレイヤのデータのソート，順位付け，表示をするクラス
Player クラス　　：プレイヤのデータを保持するクラス

4. PlayerRanking クラス，Player クラスにおけるメンバ変数とメソッドは，次のとおりである。

PlayerRanking
Player[] players　（全プレイヤのデータを保持する）
setPlayers メソッド　（配列変数から全プレイヤのデータを players へ設定する） sort メソッド　（プレイヤの情報を降順にソートする） ranking メソッド　（プレイヤの情報に順位を追加する） disp メソッド　（プレイヤの情報を表示する）

Player
int rank　（プレイヤの順位） int point　（プレイヤのポイント） String name　（プレイヤの名前）
setData メソッド　（プレイヤの情報を設定する）

≪実行例≫

```
>java Q6
Rank:1 point:51 name:Dincer
Rank:2 point:33 name:Brucs
Rank:2 point:33 name:Eric
Rank:4 point:24 name:Alex
Rank:5 point:20 name:Cyndy
```

学科名　　科　学籍番号　　氏名

問２　（つづき）

＜プログラム＞

```
 1 public class Q6 {
 2    public static void main(String[] args) {
 3       int[] points = { 24, 33, 20, 51, 33 };
 4       String[] names = { "Alex", "Brucs", "Cyndy", "Dincer", "Eric" };
 5
 6       PlayerRanking ranking = new PlayerRanking();
 7       ranking.setPlayers(points, names);
 8       ranking.sort();
 9       ranking.ranking();
10       ranking.disp();
11    }
12 }
13
14 public class PlayerRanking {
15    Player[] players;
16
17    void setPlayers(int[] points, String[] names) {
18       players = new Player[points.length];
19       for (int i = 0; i < points.length; i++) {
20          players[i] = [ (33) ];
21          players[i].setData(points[i], names[i]);
22       }
23    }
24
25    //  プレイヤの情報を降順にソートする
26    void sort() {
27       for (int i = 0; i < players.length - 1; i++) {
28          int j = i + 1;
29          Player tmp = players[j];
30          while (0 < j && [ (34) ] ) {
31             players[j] = players[j - 1];
32             j--;
33          }
34          players[j] = tmp;
35       }
36    }
37
38    //  プレイヤの情報に順位を追加する
39    void ranking() {
40       int r = 1;
41       for (int i = 1; i < players.length; i++) {
42          if (players[i - 1].point > players[i].point)
43             r = [ (35) ];
44          players[i].rank = r;
45       }
46    }
47
48    //  プレイヤの情報を表示する
49    public void disp() {
50       for (Player p : [ (36) ] ) {
51          System.out.println("Rank:" + p.rank +
52             " point:" + p.point + " name:" + p.name);
53       }
54    }
55 }
56
57 public class Player {
58    int rank;
59    int point;
60    String name;
61
62    //  プレイヤの情報を設定する
63    public void setData(int point, String name) {
64       this.point = point;
65       this.name = name;
66       rank = [ (37) ];
67    }
68 }
```

(33) の解答群

ア　Player()
イ　new Player()
ウ　Player(p.length)
エ　new Players()

(34) の解答群

ア　players[j].point > tmp.point
イ　players[j].point < tmp.point
ウ　players[j - 1].point > tmp.point
エ　players[j - 1].point < tmp.point

(35) の解答群

ア　i
イ　i + 1
ウ　players[i].rank
エ　players[i].rank + 1

(36) の解答群

ア　players
イ　players[]
ウ　players[0]
エ　ranking

(37) の解答群

ア　""
イ　null
ウ　0
エ　1

(33)	(34)	(35)	(36)	(37)
イ	エ	イ	ア	エ