選択した問題は，選択欄の(選)をマークしてください。マークがない場合は，採点されません。

問 11　次の Java プログラムの説明及びプログラムを読んで，設問 1，2 に答えよ。

（Java プログラムで使用する API の説明は，この冊子の末尾を参照してください。）

〔プログラムの説明〕

升目を用いて表現された迷路と，迷路上に置かれて外部から操作される駒を表すプログラムである。

迷路は，駒が通れる升（以下，通路という）と通れない升（以下，壁という）から成る。迷路の外周は壁である。通路のうちの一つが開始地点であり，開始地点でない通路のうちの一つがゴール地点である。本問で扱う迷路を，図 1 に示す。

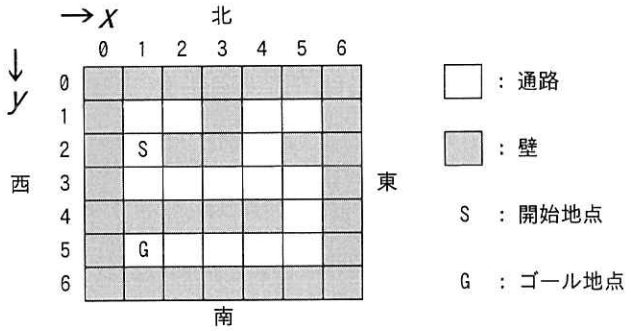

図 1　本問で扱う迷路

迷路上の升の位置は，2 次元の座標 (x, y) で表す。x と y はともに非負整数である。ある位置を基準として，x の値が大きくなる方角を東，小さくなる方角を西，y の値が大きくなる方角を南，小さくなる方角を北とする。

駒は，東西南北のいずれかを向いており，向いている方角を基準として，次の三つの操作を外部から受け付ける。

① 左の方向に向きを変える。

② 右の方向に向きを変える。

③ 隣接する前方の升が通路なら，1 升前進する。

(1) クラス Maze は迷路を表す。コンストラクタの引数には，文字列で表現した迷路と，迷路の西端から東端までの升の個数を指定する。迷路を表現する文字列は，1升を表す文字を西から東に向かって順に並べた1行分の文字列を，北から南に向かって順に連結したものである。升の種類は，char 型の値で表す。“*”は壁を，それ以外の値は通路を表し，“S”は開始地点を，“G”はゴール地点を表す。引数に誤りはないものとする。

メソッド getStartLocation は開始地点の座標を返す。メソッド isGoal は指定された座標の升がゴール地点であれば true を，それ以外は false を返す。メソッド isBlank は指定された座標の升が通路ならば true を，壁ならば false を返す。

(2) クラス Piece は，迷路上に置かれる駒を表す。コンストラクタの引数で迷路を指定する。インスタンスは，最初は，開始地点に位置し，北を向いている。

メソッド turnLeft は左の方向に，turnRight は右の方向に向きを変える。

メソッド tryStepForward は，隣接する前方の升が通路なら1升前進し，前進した方角を履歴リストに追加してから true を返す。通路でなければ，前進せずに false を返す。

メソッド isAtGoal は，ゴール地点にいれば true を，それ以外は false を返す。

メソッド getHistory は，履歴リストを返す。

(3) 列挙 Direction は，方角を表す。

メソッド left は列挙定数が表す方角に向かって左の方角を，right は右の方角を返す。

(4) クラス Location は，迷路上の升の位置を示す座標を表す。

(5) クラス PlayMaze は，図1に示す開始地点からゴール地点に至るまで駒を操作し，その後，履歴リストを表示する。

〔プログラム1〕

```
public class Maze {
   private final String mazeData;
   private final int width;
   private final Location startLocation;

   public Maze(String mazeData, int width) {
      this.mazeData = mazeData;
```

```
        this.width = width;
        startLocation = locationOf('S');
    }

    public Location getStartLocation() { return startLocation; }

    public boolean isGoal(Location loc) {
        return mazeData.charAt(loc.y [ a ] width + loc.x) == 'G';
    }

    public boolean isBlank(Location loc) {
        return mazeData.charAt(loc.y [ a ] width + loc.x) != '*';
    }

    private Location locationOf(char c) {
        int index = mazeData.indexOf(c);
        return new Location(index [ b ] width, index / width);
    }
}
```

〔プログラム2〕

```
import java.util.ArrayList;
import java.util.List;

public class Piece {
    private final Maze maze;
    private Location location;
    private Direction direction = Direction.NORTH;
    private final List<Direction> history = new ArrayList<>();

    public Piece(Maze maze) {
        this.maze = maze;
        location = maze.getStartLocation();
    }

    public void turnLeft() { direction = direction.left(); }

    public void turnRight() { direction = direction.right(); }

    public boolean tryStepForward() {
        Location nextLocation = new Location( [ c ] );
        if (maze.isBlank(nextLocation)) {
```

```
            location = nextLocation;
            history.add(direction);
            return true;
        }
        return false;
    }

    public boolean isAtGoal() { return maze.isGoal(location); }

    public List<Direction> getHistory() { return new ArrayList<>(history); }
}
```

〔プログラム 3〕

```
public enum Direction {
    NORTH(0, -1), EAST(1, 0), SOUTH(0, 1), WEST(-1, 0);

    public int dx, dy;

    private Direction(int dx, int dy) {
        this.dx = dx;
        this.dy = dy;
    }

    // クラスメソッド values は，この列挙で定義している列挙定数を，
    // 定義順に格納した配列を返す。
    // メソッド ordinal は，この列挙定数の定義順（先頭は 0）を返す。
    public Direction left() { return values()[ d ]; }

    public Direction right() { return values()[(ordinal() + 1) % 4]; }
}
```

〔プログラム 4〕

```
public class Location {
    public final int x, y;

    public Location(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

〔プログラム5〕

```
import java.util.List;

public class PlayMaze {
   public static void main(String... args) {
      Maze maze = new Maze("*******" +
                           "*..*..*" +
                           "*S**.**" +
                           "*.....*" +
                           "*****.*" +
                           "*G....*" +
                           "*******", 7);
      Piece piece = new Piece(maze);
      while (!piece.isAtGoal()) {
         piece.turnLeft();
         while (!piece.tryStepForward()) {
            piece.turnRight();
         }
      }
      List<Direction> history = piece.getHistory();
      System.out.println(history);
   }
}
```

α

Java

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

a，bに関する解答群

ア　%　　イ　&　　ウ　*　　エ　+

オ　-　　カ　/　　キ　^　　ク　|

cに関する解答群

ア　direction

イ　direction.dx, direction.dy

ウ　location + direction

エ　location.x + direction.dx, location.y + direction.dy

dに関する解答群

ア　(ordinal() + 3) % 4　　イ　(ordinal() + 3) / 4

ウ　(ordinal() - 1) % 4　　エ　(ordinal() - 1) / 4

設問2　プログラム5のαの位置に次の処理を挿入し，実行結果として，図2に示す方角のリストを得た。駒は，開始地点からリストの方角の順に1升ずつ進むと，直前の升に戻る（正反対の方角に向きを変えて進む）ことなく，ゴール地点に至ることができる。［　　］に入れる正しい答えを，解答群の中から選べ。

```
for (int i = [ e ]; i < history.size(); i++) {
   if (history.get(i - 1) == history.get(i).left().left()) {
      history.remove([ f ]);
      history.remove([ f ]);
      i = i < 2 ? 0 : i - 2;
   }
}
```

```
[SOUTH, EAST, EAST, EAST, EAST, SOUTH, SOUTH, WEST, WEST, WEST, WEST]
```

図2　方角のリスト

eに関する解答群

ア　-1　　イ　0　　ウ　1

fに関する解答群

ア　i　　イ　i + 1　　ウ　i - 1