Past-paper walk-through

while Loops ▪ 2.7

eXercise

Questions in this set

- 2024 Q1
- 2024 Q4
- 2018 Q1

Reading the mark scheme

- **B1** a mark that stands on its own – the point is either made or not.
- **M1** a *method* mark: the right approach, even if the number is wrong.
- **A1** an *answer* mark, and it usually needs its M mark first.
- **C1** a working mark on the way to the answer.
- // separates answers the examiner accepts equally.
- **ecf** = error carried forward: a later part is marked on your own earlier answer.
- **owtte** = "or words to that effect" – the wording need not match.

Question 1

2024 ▪ Q1

This question simulates birds or possibly a bear eating at a bird feeder. The following Feeder class updates information about how much food is in the bird feeder and simulates how much food is eaten. You will write two methods of the Feeder class.

```
k+kdpublic+w k+kdclass n+ncFeeder
p
+w    c+cm/**
c+cm     * The amount of food, in grams, currently in the bird feeder; initialized in the constructor
c+cm     * always greater than or equal to zero
c+cm     */
+w    k+kdprivate+w k+ktint+w ncurrentFoodp;

+w    c+cm/**
c+cm     * Simulates one day with numBirds birds or possibly a bear at the bird feeder,
c+cm     * as described in part (a)
c+cm     * Precondition: numBirds  0
c+cm     */
+w    k+kdpublic+w k+ktvoid+w n+nfsimulateOneDayp(k+ktint+w nnumBirdsp)
+w    p+w  c+cm/* to be implemented in part (a) */+w  p
+w    c+cm/**
```

Question 1

(a) Write the `simulateOneDay` method, which simulates `numBirds` birds or possibly a bear at the bird feeder for one day. The method determines the amount of food taken from the feeder on this day and updates the `currentFood` instance variable. The simulation accounts for normal conditions, which occur 95% of the time, and abnormal conditions, which occur the other 5% of the time.

Under normal conditions, the simulation assumes that on any given day, only birds visit the feeder and that each bird at the feeder consumes the same amount of food. This standard amount consumed is between 10 and 50 grams of food, inclusive, in 1-gram increments. That is, on any given day, each bird might eat 10, 11, . . . , 49, or 50 grams of food. The amount of food eaten by each bird on a given day is randomly generated and each integer from 10 to 50, inclusive, has an equal chance of being chosen.

Question 1

For example, a run of the simulation might predict that for a certain day under normal conditions, each bird coming to the feeder will eat 11 grams of food. If 10 birds come to the feeder on that day, then a total of 110 grams of food will be consumed.
If the simulated food consumed is greater than the amount of food in the feeder at the beginning of the day, the simulation empties the feeder and the amount of food in the feeder at the end of the day is zero.
Under abnormal conditions, a bear empties the feeder and the amount of food in the feeder at the end of the day is zero.
The following examples show possible results of three calls to `simulateOneDay`.

- Example 1: If the feeder initially contains 500 grams of food, the call `simulateOneDay(12)` could result in 12 birds eating 20 grams of food each, leaving 260 grams of food in the feeder.
- Example 2: If the feeder initially contains 1,000 grams of food, the call `simulateOneDay(22)` could result in a bear eating all the food, leaving 0 grams of food in the feeder.
- Example 3: If the feeder initially contains 100 grams of food, the call `simulateOneDay(5)` could result in 5 birds attempting to eat 30 grams of food each. Since the feeder initially contains less than 150 grams of food, the feeder is emptied, leaving 0 grams of food in the feeder.

Question 1

Complete the `simulateOneDay` method.

```
c+cm/**
c+cm * Simulates one day with numBirds birds or possibly a bear at the bird feede
c+cm * as described in part (a)
c+cm * Precondition: numBirds  0
c+cm */
k+kdpublic+w k+ktvoid+w n+nfsimulateOneDayp(k+ktint+w nnumBirdsp)
```

Question 1

Class information for this question

```
k+kdpublic+w k+kdclass n+ncFeeder
k+kdprivate+w k+ktint+w ncurrentFood
k+kdpublic+w k+ktvoid+w n+nfsimulateOneDayp(k+ktint+w nnumBirdsp)
k+kdpublic+w k+ktint+w n+nfsimulateManyDaysp(k+ktint+w nnumBirdsp,+w k+kti
```

Question 1

2024 ▪ Q1

This question simulates birds or possibly a bear eating at a bird feeder. The following `Feeder` class updates information about how much food is in the bird feeder and simulates how much food is eaten. You will write two methods of the `Feeder` class.

```
k+kdpublic+w k+kdclass n+ncFeeder
p
+w    c+cm/**
c+cm     * The amount of food, in grams, currently in the bird feeder; initialized in the constructor
c+cm     * always greater than or equal to zero
c+cm     */
+w    k+kdprivate+w k+ktint+w ncurrentFoodp;

+w    c+cm/**
c+cm     * Simulates one day with numBirds birds or possibly a bear at the bird feeder,
c+cm     * as described in part (a)
c+cm     * Precondition: numBirds  0
c+cm     */
+w    k+kdpublic+w k+ktvoid+w n+nfsimulateOneDayp(k+ktint+w nnumBirdsp)
+w    p+w  c+cm/* to be implemented in part (a) */+w  p
```

Question 1

(b) Write the `simulateManyDays` method. The method uses `simulateOneDay` to simulate the feeder on at most `numDays` consecutive days. The `numBirds` birds or a bear coming to the feeder on each day of the simulation returns the number of days that birds or a bear found food at the feeder.

Consider the following examples.

Value of `currentFood` and Method Call	Possible Outcomes and Resulting Return Value
`currentFood` = 2400 `simulateManyDays(610, 4)`	Day 1: `simulateOneDay` leaves 2100 grams of food in the feeder. Day 2: `simulateOneDay` leaves 1650 grams of food in the feeder. Day 3: `simulateOneDay` leaves 1500 grams of food in the feeder. Day 4: `simulateOneDay` leaves 1260 grams of food in the feeder. The simulation returns 4 because, on all four days of the simulation, birds or a bear found food at the feeder. The instance variable `currentFood` has the value 1260.
`currentFood` = 250 `simulateManyDays(150, 5)`	Day 1: `simulateOneDay` leaves 150 grams of food in the feeder. Day 2: `simulateOneDay` leaves 0 grams of food in the feeder. The simulation returns 2 because, on two of the five simulated days, birds or a bear found food at the feeder. The instance variable `currentFood` has the value 0.
`currentFood` = 0 `simulateManyDays(10, 3)`	The simulation returns 0 because no food was found at the feeder on any day.

Question 1

Complete the `simulateManyDays` method. Assume that `simulateOneDay` works as intended, regardless of what you wrote in part (a). You must use `simulateOneDay` appropriately in order to receive full credit.

```
c+cm/**
c+cm * Returns the number of days birds or a bear found food to eat at the feeder
c+cm * as described in part (b)
c+cm * Precondition: numBirds  0, numDays  0
c+cm */
k+kdpublic+w k+ktint+w n+nfsimulateManyDaysp(k+ktint+w nnumBirdsp,+w k+ktint+w nnu
```

Question 1

Class information for this question

```
k+kdpublic+w k+kdclass n+ncFeeder
k+kdprivate+w k+ktint+w ncurrentFood
k+kdpublic+w k+ktvoid+w n+nfsimulateOneDayp(k+ktint+w nnumBirdsp)
k+kdpublic+w k+ktint+w n+nfsimulateManyDaysp(k+ktint+w nnumBirdsp,+w k+kti
```

Question 4

2024 ▪ Q4

This question involves a path through a two-dimensional (2D) array of integers, where the path is based on the values of elements in the array. When an element of the 2D array is accessed, the first index is used to specify the row and the second index is used to specify the column. The following `Location` class represents a row and column position in the 2D array.

```
k+kdpublic+w k+kdclass n+ncLocation
p
+w    k+kdprivate+w k+ktint+w ntheRowp;
+w    k+kdprivate+w k+ktint+w ntheColp;

+w    k+kdpublic+w n+nfLocationp(k+ktint+w nrp,+w k+ktint+w ncp)
+w    p
+w        ntheRow+w o=+w nrp;
+w        ntheCol+w o=+w ncp;
+w    p
+w    k+kdpublic+w k+ktint+w n+nfgetRowp(p)
+w    p+w  kreturn+w ntheRowp;+w  p

+w    k+kdpublic+w k+ktint+w n+nfgetColp(p)
+w    p+w  kreturn+w ntheColp;+w  p
```

Question 4

The following GridPath class contains the 2D array and methods to use to determine a path through the array. You will write two methods of the GridPath class.

```
k+kdpublic+w k+kdclass n+ncGridPath
p
+w    c+cm/** Initialized in the constructor with distinct values that never change */
+w    k+kdprivate+w k+ktinto[o]o[o]+w ngridp;

+w    c+cm/**
c+cm     * Returns the Location representing a neighbor of the grid element at row and col,
c+cm     * as described in part (a)
c+cm     * Preconditions: row is a valid row index and col is a valid column index in grid.
c+cm     *      row and col do not specify the element in the last row and last column of grid.
c+cm     */
+w    k+kdpublic+w nLocation+w n+nfgetNextLocp(k+ktint+w nrowp,+w k+ktint+w ncolp)
+w    p+w  c+cm/* to be implemented in part (a) */+w  p
+w    c+cm/**
c+cm     * Computes and returns the sum of all values on a path through grid, as described in
c+cm     * part (b)
c+cm     * Preconditions: row is a valid row index and col is a valid column index in grid.
c+cm     *      row and col do not specify the element in the last row and last column of grid.
```

Question 4

(a) Write the `getNextLoc` method, which returns a `Location` object that represents the smaller of two neighbors of the `grid` element at `row` and `col`, according to the following rules.

- The two neighbors that are considered as the element below the given element and the element to the right of the given element, if they exist.
- If both neighbors exist, the `Location` of the neighbor with the smaller value is returned. Two neighbors will always have different values.
- If only one neighbor exists, the `Location` of the existing neighbor is returned.

For example, assume that `grid` contains the following values.

	0	**1**	**2**	**3**	**4**
0	12	3	4	13	5
1	11	21	2	14	16
2	7	8	9	15	0
3	10	17	20	19	1
4	18	22	30	25	6

Question 4

The following table shows some sample calls to `getNextLoc`.

Method Call	Explanation
`getNextLoc(0, 0)`	Returns the neighbor to the right (the `Location` representing the element at row 0 and column 1), since $3 < 11$.
`getNextLoc(1, 3)`	Returns the neighbor below (the `Location` representing the element at row 2 and column 3), since $15 < 16$.
`getNextLoc(2, 4)`	Returns the neighbor below (the `Location` representing the element at row 3 and column 4), since the given element has no neighbor to the right.
`getNextLoc(4, 3)`	Returns the neighbor to the right (the `Location` representing the element at row 4 and column 4), since the given element has no neighbor below.

Question 4

In the example, the `getNextLoc` method will never be called with row 4 and column 4, as those values would violate the precondition of the method.
Complete the `getNextLoc` method.

```
c+cm/**
c+cm * Returns the Location representing a neighbor of the grid element at row and col,
c+cm * as described in part (a)
c+cm * Preconditions: row is a valid row index and col is a valid column index in grid.
c+cm *      row and col do not specify the element in the last row and last column of grid
c+cm */
k+kdpublic+w nLocation+w n+nfgetNextLocp(k+ktint+w nrowp,+w k+ktint+w ncolp)
```

Question 4

Class information for this question

```
k+kdpublic+w k+kdclass n+ncLocation
k+kdprivate+w k+ktint+w ntheRow
k+kdprivate+w k+ktint+w ntheCol

k+kdpublic+w n+nfLocationp(k+ktint+w nrp,+w k+ktint+w ncp)
k+kdpublic+w k+ktint+w n+nfgetRowp(p)
k+kdpublic+w k+ktint+w n+nfgetColp(p)

k+kdpublic+w k+kdclass n+ncGridPath
k+kdprivate+w k+ktinto[o]o[o]+w ngrid

k+kdpublic+w nLocation+w n+nfgetNextLocp(k+ktint+w nrowp,+w k+ktint+w ncolp)
k+kdpublic+w k+ktint+w n+nfsumPathp(k+ktint+w nrowp,+w k+ktint+w ncolp)
```

Question 4

	0	1	2	3	4
0	12	3	4	13	5
1	11	21	2	14	16
2	7	8	9	15	0
3	10	17	20	19	1
4	18	22	30	25	6

Question 4

	0	1	2	3	4
0	12	30	40	25	5
1	11	3	22	15	43
2	7	2	9	4	0
3	8	33	18	6	1

Question 4

2024 ▪ Q4

This question involves a path through a two-dimensional (2D) array of integers, where the path is based on the values of elements in the array. When an element of the 2D array is accessed, the first index is used to specify the row and the second index is used to specify the column. The following `Location` class represents a row and column position in the 2D array.

```
k+kdpublic+w k+kdclass n+ncLocation
p
+w    k+kdprivate+w k+ktint+w ntheRowp;
+w    k+kdprivate+w k+ktint+w ntheColp;

+w    k+kdpublic+w n+nfLocationp(k+ktint+w nrp,+w k+ktint+w ncp)
+w    p
+w        ntheRow+w o=+w nrp;
+w        ntheCol+w o=+w ncp;
+w    p

+w    k+kdpublic+w k+ktint+w n+nfgetRowp(p)
+w    p+w  kreturn+w ntheRowp;+w  p
```

Question 4

The following GridPath class contains the 2D array and methods to use to determine a path through the array. You will write two methods of the GridPath class.

```
k+kdpublic+w k+kdclass n+ncGridPath
p
+w    c+cm/** Initialized in the constructor with distinct values that never change */
+w    k+kdprivate+w k+ktinto[o]o[o]+w ngridp;

+w    c+cm/**
c+cm     * Returns the Location representing a neighbor of the grid element at row and col,
c+cm     * as described in part (a)
c+cm     * Preconditions: row is a valid row index and col is a valid column index in grid.
c+cm     *      row and col do not specify the element in the last row and last column of grid.
c+cm     */
+w    k+kdpublic+w nLocation+w n+nfgetNextLocp(k+ktint+w nrowp,+w k+ktint+w ncolp)
+w    p+w  c+cm/* to be implemented in part (a) */+w  p
+w    c+cm/**
c+cm     * Computes and returns the sum of all values on a path through grid, as described in
c+cm     * part (b)
c+cm     * Preconditions: row is a valid row index and col is a valid column index in grid.
c+cm     *      row and col do not specify the element in the last row and last column of grid.
```

Question 4

(b) Write the `sumPath` method, which returns the sum of all values on a path in `grid`. The path begins with the element at `row` and `col` and is determined by successive calls to `getNextLoc`. The path ends when the element in the last row and last column of `grid` is reached.
For example, consider the following contents of `grid`. The shaded elements of `grid` represent the values on the path that results from the method call `sumPath(1, 1)`. The method call returns 19 because $3 + 2 + 9 + 4 + 0 + 1 = 19$.

	0	**1**	**2**	**3**	**4**
0	12	30	40	25	5
1	11	**3**	22	15	43
2	7	**2**	**9**	**4**	**0**
3	8	33	18	6	**1**

Question 4

[Grid diagram: 4x5 table of integers, rows 0-3 and columns 0-4 labelled. The shaded (bolded) path cells forming the path from sumPath(1,1) are: (1,1)=3, (2,1)=2, (2,2)=9, (2,3)=4, (2,4)=0, (3,4)=1, matching the sum $3 + 2 + 9 + 4 + 0 + 1 = 19$.]
Write the `sumPath` method. Assume `getNextLoc` works as intended, regardless of what you wrote in part (a). You must use `getNextLoc` appropriately in order to receive full credit.

```
c+cm/**
c+cm * Computes and returns the sum of all values on a path through grid, as described in
c+cm * part (b)
c+cm * Preconditions: row is a valid row index and col is a valid column index in grid.
c+cm *      row and col do not specify the element in the last row and last column of grid
c+cm */
k+kdpublic+w k+ktint+w n+nfsumPathp(k+ktint+w nrowp,+w k+ktint+w ncolp)
```

Question 4

Class information for this question

```
k+kdpublic+w k+kdclass n+ncLocation
k+kdprivate+w k+ktint+w ntheRow
k+kdprivate+w k+ktint+w ntheCol

k+kdpublic+w n+nfLocationp(k+ktint+w nrp,+w k+ktint+w ncp)
k+kdpublic+w k+ktint+w n+nfgetRowp(p)
k+kdpublic+w k+ktint+w n+nfgetColp(p)

k+kdpublic+w k+kdclass n+ncGridPath
k+kdprivate+w k+ktinto[o]o[o]+w ngrid

k+kdpublic+w nLocation+w n+nfgetNextLocp(k+ktint+w nrowp,+w k+ktint+w ncolp)
k+kdpublic+w k+ktint+w n+nfsumPathp(k+ktint+w nrowp,+w k+ktint+w ncolp)
```