Started on	Wednesday, 29 March 2023, 18:14
State	Finished
Completed on	Wednesday, 29 March 2023, 18:15
Time taken	1 min 54 secs
Grade	**7.00** out of 7.00 (**100%**)

Question **1**

Correct

Mark 1.00 out of 1.00

Assume I have a function f(int K) that runs in **amortized logarithmic time** in K, but **linear worst case time**. What is the running time of:

```
# python 3
for i in range(N):
    f(N)
```

```
// java
for (int i = 0; i < N; i=i+1)
    f(N)
```

Select one:

- a. Impossible to say from the information given
- b. Linear in N
- c. Linerarithmic in N ✓
- d. Quadratic in N

Your answer is correct.

The correct answer is: Linerarithmic in N

Question 2

Correct

Mark 1.00 out of 1.00

Assume I have a function f(int K) that runs in **amortized constant time** in K, but **logarithmic worst case time**. What is the running time of

```
# python 3
for i in range(N):
    f(N)
```

```
// java
for (int i = 0; i < N; i=i+1)
    f(N)
```

Select one:

- a. Linearithmic in N
- b. Linear in N ✓
- c. Quadratic in N
- d. Impossible to say from the information given

Your answer is correct.

The correct answer is: Linear in N

Question 3

Correct

Mark 1.00 out of 1.00

Assume I have a function f(int K) that runs in **amortized constant time** in K, but **linear worst case time**. What is the running time of

```
# python 3
for i in range(N):
    f(N)
```

```
// java
for (int i = 0; i < N; i=i+1)
    f(N)
```

Select one:

- a. Impossible to say from the information given
- b. Quadratic in N
- c. Linear in N ✓
- d. Linearithmic in N

Your answer is correct.

The correct answer is: Linear in N

Information

For the questions on this page consider below class *Y*:

```
public class Y<Key extends Comparable<Key>>
{
  private Key[] A = (Key[]) new Comparable[1];
  private int lo, hi, N;

  public void insert(Key in)
  {
    A[hi] = in;
    hi = hi + 1;
    if (hi == A.length) hi = 0;
    N = N + 1;
    if (N == A.length) rebuild();
  }

  public Key remove() // assumes Y is not empty
  {
    Key out = A[lo];
    A[lo] = null;
    lo = lo + 1;
    if (lo == A.length) lo = 0;
    N = N - 1;
    return out;
  }

  private void rebuild()
  {
    Key[] tmp = (Key[]) new Comparable[2*A.length];
    for (int i = 0; i < N; i++ )
    tmp[i] = A[(i + lo)  % A.length];
    A = tmp;
    lo = 0;
    hi = N;
  }
}
```

Question 4

Correct

Mark 1.00 out of 1.00

What is the amortized number of array accesses per operation in a sequence of *k* Y.insert operations beginning in an empty data structure?

Select one:

- a. Constant ✓
- b. Quadratic in K
- c. Linearithmic in K
- d. Linear in K

Your answer is correct.

The correct answer is: Constant

Question 5

Correct

Mark 1.00 out of 1.00

What is a best bound on the number of array accesses **per operation** in the following sequence of 2k operations, starting from an empty data structure:

```
y.insert(1);
y.remove();
y.insert(2);
y.remove();
y.insert(3);
y.remove();
 . . .
y.insert(k);
y.remove();
```

Select one:

- a. linear in *k* in the worst case and in the amortized sense
- b. Constant in the worst case ✓
- c. Quadratic in the worst case
- d. Constant in the amortized sense, but linear in the worst case

Your answer is correct.

The correct answer is: Constant in the worst case

Information

For the questions on this page consider the following class:

```
public class X<Key extends Comparable<Key>>
{
    private Key[] A = (Key[]) new Comparable[1];
    private int N = 0;

    public void insert(Key in)
    {
        A[N] = in;
        N = N + 1;
        if (N == A.length) rebuild();
    }

    public Key remove() // assumes X is not empty
    {
        // search for maximum key:
        int max = 0;
        for (int i = 1; i < N; i++)
          if (A[i].compareTo(A[max]) > 0)
            max = i;
        // exchange with last element:
        Key tmp = A[max];
        A[max] = A[N-1];

        N = N - 1;
        return tmp;
    }

    private void rebuild()
    {
        Key[] tmp = (Key[]) new Comparable[2*A.length];
        for (int i = 0; i < N; i++ )
            tmp[i] = A[i];
        A = tmp;
    }
}
```

Question 6

Correct

Mark 1.00 out of 1.00

What is the amortized number of array accesses per operation in a sequence of k insert operations beginning in an empty data structure?

Select one:

- a. Linear in k
- b. Constant ✓
- c. Linearithmic in k
- d. Quadratic in k

Your answer is correct.

The correct answer is: Constant

Question 7

Correct

Mark 1.00 out of 1.00

What is the amortized number of array accesses per operation in a sequence of k remove operations beginning in a data structure with k elements?

Select one:

- a. Linearithmic in k
- b. Constant
- c. Quadratic in k
- d. Linear in k ✓

Your answer is correct.

The correct answer is: Linear in k