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。

〔プログラムの説明〕

クラス ArrayAppendableCharSequence 及び ListAppendableCharSequence は，追加可能な0個以上の文字からなる文字列（文字の並び）を表現するインタフェース AppendableCharSequence を異なるデータ構造で実装したプログラムである。

インタフェース AppendableCharSequence は，パッケージ java.lang のインタフェース CharSequence 及び Appendable で定義されている次のメソッドからなる。

(1) メソッド charAt は，引数 index で指定された位置の文字を返す。文字の位置はインデックス値で表され，文字列の長さが1以上のとき，最初の文字の位置は0，最後の文字の位置は文字列の長さ－1で表される。引数で指定されたインデックス値が負又は文字列の長さ以上の場合は，IndexOutOfBoundsException を投げる。

(2) メソッド length は，文字列の長さを返す。

(3) メソッド append は，引数で指定された文字を文字列の末尾に追加する。戻り値として，この AppendableCharSequence のインスタンス自身を返す。

(4) メソッド toString は，このインスタンスの文字列を String オブジェクトで返す。

クラス ArrayAppendableCharSequence は，char 型の配列を用いてインタフェース AppendableCharSequence を実装したものである。メソッド append は，配列の空いている最初の要素に文字を格納する。配列に空き要素がないときは，要素の個数が現在の配列よりも EXT_SIZE の値だけ大きな配列を生成し，既存の文字データを移し，空いている最初の要素に文字を格納する。

クラス ListAppendableCharSequence は，連結リスト構造を用いてインタフェース AppendableCharSequence を実装したものである。入れ子クラス ListAppendableCharSequence.Bucket（以下，Bucket という）は，連結リストの要素である。Bucket のインスタンス1個につき，要素の個数が EXT_SIZE である char 型の配列を用意し，そこに文字データを保持する。Bucket の配列に空き要素がなくなり，次の文字を追加できないときは，新規に Bucket のインスタンスを生成し，連結リストに追加する。

クラス Test は，上記二つのクラスそれぞれの実装が異なるメソッド append の実

行時間を測定するプログラムである。メソッド measureTime は，引数 n で指定された回数だけ，引数 a で指定されたインスタンスのメソッド append を呼び出し，実行時間をミリ秒単位で測定する。

〔プログラム1〕

```
public interface AppendableCharSequence {
   public char charAt(int index);
   public int length();
   public AppendableCharSequence append(char c);
   public String toString();
}
```

〔プログラム2〕

```
public class ArrayAppendableCharSequence
             implements AppendableCharSequence {
   private static final int EXT_SIZE = 10;
   private int length;
   private char[] data;

   public ArrayAppendableCharSequence() {
      data = new char[EXT_SIZE];
   }

   public char charAt(int index) {
      if (index < 0 || index >= length)
         throw new IndexOutOfBoundsException();
      return data[index];
   }

   public int length() {
      return length;
   }

   public AppendableCharSequence append(char c) {
      if (data.length == length) {
         char[] temp = new char[length + EXT_SIZE];
         for (int i = 0; i < [ a ]; i++) {
             temp[i] = data[i];
         }
         data = temp;
      }
      data[length++] = c;
      return this;
   }
```

```
    public String toString() {
       return new String(data, 0, length);
    }
  }
```

〔プログラム3〕

```
public class ListAppendableCharSequence
              implements AppendableCharSequence {
   private static final int EXT_SIZE = 10;
   private Bucket bucketList;
   private int length;

   public ListAppendableCharSequence() {
      bucketList = new Bucket();
   }

   public char charAt(int index) {
      if (index < 0 || index >= length)
         throw new IndexOutOfBoundsException();
      Bucket bucket = getBucket(index);
      return bucket.data[    b    ];
   }

   public int length() {
      return length;
   }

   public AppendableCharSequence append(char c) {
      int offset = length % EXT_SIZE;
      Bucket bucket = getBucket(    c    );
      if (offset == 0 && length != 0) {
         bucket.next = new Bucket();
         bucket = bucket.next;
      }
      bucket.data[offset] = c;
      length++;
      return this;
   }

   public String toString() {
      char[] data = new char[length];
      Bucket bucket = bucketList;
      int size;
      for (int len = length; len > 0; len -= size) {
         size = (len >= EXT_SIZE) ? EXT_SIZE : len;
         for (int i = 0, j =    d    ; i < size; i++, j++) {
             data[j] = bucket.data[i];
         }
```

```
            bucket = bucket.next;
        }
        return new String(data);
    }

    private Bucket getBucket(int index) {
        int n = index / EXT_SIZE;
        Bucket b = bucketList;
        for (int i = 0; i < n; i++) {
            b = b.next;
        }
        return b;
    }

    private static class Bucket {
        private Bucket next;
        private char[] data = new char[EXT_SIZE];
    }
}
```

〔プログラム4〕

```
public class Test {
    static final int[] TIMES = {
        5000, 10000, 50000, 100000
    };

    public static void main(String[] args) {
        for (int n : TIMES) {
            measureTime(n, new ArrayAppendableCharSequence());
            measureTime(n, new ListAppendableCharSequence());
        }
    }

    static void measureTime(int n, [ e ] a) {
        long start = System.currentTimeMillis();
        for (int i = 0; i < n; i++) {
            a.append((char) (i % 26 + 'a'));
        }
        long end = System.currentTimeMillis();
        System.out.printf("%s: %d [times] %d [ms]%n",
                          a.getClass().getName(),
                          n, end - start);
    }
}
```

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に関する解答群

ア　data.length + length　　イ　data.length + temp.length

ウ　length　　エ　length + EXT_SIZE

オ　temp.length

bに関する解答群

ア　index　　イ　index % EXT_SIZE

ウ　index / EXT_SIZE　　エ　length

オ　length % EXT_SIZE　　カ　length / EXT_SIZE

cに関する解答群

ア　length　　イ　length + 1　　ウ　length - 1

エ　offset　　オ　offset + 1　　カ　offset - 1

dに関する解答群

ア　len　　イ　len % EXT_SIZE　　ウ　len - length

エ　length　　オ　length % EXT_SIZE　　カ　length - len

eに関する解答群

ア　AppendableCharSequence

イ　ArrayAppendableCharSequence

ウ　ListAppendableCharSequence

エ　ListAppendableCharSequence.Bucket

オ　Object

カ　String

設問2　クラス Test を用いて各クラスのメソッド append の実行時間（ミリ秒）を測定したところ，表1の結果になった。

表1　各クラスのメソッド append の実行時間

単位　ミリ秒

呼出し回数 / クラス名	5,000回	10,000回	50,000回	100,000回
ArrayAppendableCharSequence	5	18	370	1,455
ListAppendableCharSequence	4	8	960	4,085

文字の追加回数が多くなるにつれて ListAppendableCharSequence の方が遅くなるのは，連結リストをたどる回数が増えるためであると推測される。そこで，ListAppendableCharSequence.append の性能を改善するために，次の修正を行った。［　　］に入れる正しい答えを，解答群の中から選べ。

なお，ほかの変更はないものとする。

(1)　連結リストの最後の要素を常に参照するためのフィールド last を追加する。

```
private Bucket last;
```

(2)　コンストラクタでフィールド last を初期化する。

```
public ListAppendableCharSequence() {
   bucketList = new Bucket();
   last = [    f    ];
}
```

(3)　メソッド append でフィールド last を利用する。

```
public AppendableCharSequence append(char c) {
   int offset = length % EXT_SIZE;
   if (offset == 0 && length != 0) {
      last.next = new Bucket();
      last = [    g    ];
   }
   last.data[offset] = c;
   length++;
   return this;
}
```

解答群

ア　bucketList	イ　bucketList.next	ウ　last
エ　last.next	オ　new Bucket()	カ　null