```
public class MidtermLongAnswer{

   public static void main(String[] args){

      String test = "belloo 123";
      int score = scoreString(test);
      System.out.println(test + " has score: " + score); //95

      String[] sArr = {"hello", "belloo 123", "aaaaaaaaaaaa",
      "COMP 202 is lots of fun!!!      ", "bmgbmgbmgbmg", " x x x x x x "};

      for (int i=0; i < sArr.length; i++){
         score = scoreString(sArr[i]);
         System.out.println(sArr[i] + " has score: " + score);
      }

      String maxString = findMaxString(sArr);
      System.out.println("Max String is: " + maxString);
   }
//    Write the scoreChar method. This method takes a character and returns the characters score as an
integer.
//    A character's score is determined by the rules in the following table:
   public static int scoreChar(char c){
      //b, m, or g
      //(lowercase only)   5
      if (c == 'b' || c == 'm' || c == 'g'){
         return 5;
      }

      //Any digit (0,1...9)   10
```

```
    else if ('0' <= c && c <= '9'){
        return 10;
    }
    //A space
    else if (c == ' '){
        return -10;
    }
    //Anything else
    else{
        return 0;
    }
}

//Write the scoreString method. This method takes a String and returns a score for the String as an
//integer. This method must not print anything.
public static int scoreString(String s){
    int score = 0;

    for (int i=0; i < s.length(); i++){

        //The sum of all the scores for each of the characters in the String
        char letter = s.charAt(i);
        score += scoreChar(letter);

        //avoid accessing a letter out of bounds
        if (i < s.length() - 1)
        {
            //Add 30 whenever there are two of the same characters in a row
            char nextLetter = s.charAt(i + 1);
            if (letter == nextLetter){
```

```
            score += 30;
          }
        }
      }
      //Finally, add the length of the String to the score
      score += s.length();
      return score;
   }

//    Write a method findMaxString. This method takes as input an array of Strings and returns the String
//    with the highest score. Note that this method must use the scoreString method to score each String.
//    Assume that the input array holds at least one String. If there is a tie in scores, return the String with
that
//    score which is earliest in the array.

   //Many different solutions here
   public static String findMaxString(String[] arr){

      //optional check for empty arr
      if (arr == null || arr.length == 0){
         return null;
      }

      //set the first String to be the max String
      //we know the arr has at least one String
      String maxString = arr[0];
      int maxScore = scoreString(maxString);

      //maxString will hold the best String we've seen so far
      //maxScore holds the best score we've seen
```

```
//other possibilities:
// - keep the index of the best String
// - keep only the max String, and repeatedly get its
// score to check against the other String

//warning: a few solutions set the maxScore to 0
//this is not a safe assumption
//maybe the input array has only Strings with negative scores
//(we didn't take marks off for this)

for (int i = 1; i < arr.length; i++){
   String toScore = arr[i];
   int score = scoreString(toScore);

   //see if this String has a better score
   if (score > maxScore){

      //if this is a better String,
      //update the max variables
      maxScore = score;
      maxString = toScore;
   }

   //note that because we only update the max variables
   //when the score is greater than the maxScore
   //the String kept in maxString will be the first
   //String with the best score

   //however, it is also fine to loop back through the
```

```
            //String array and return the first String with that
            //maxScore
        }

        return maxString;
    }
}
```