1

(A)

```
Public class RandomStringChooser[
Private boolean[]  recorder;
Private String[]  wordList;

Public RandomStringChooser (String[]  list)[
 this.wordList = list
 this.recorder = new boolean[list.length]

]//end constructor method

 Public String getNext ( ) [
 int roll = (int) (Math.random( ) * this.wordList.length + 1 )
 if(!recorder[roll]) [
 return WordList[roll]
 recorder[roll] = true;
]else[
 return "None";
  ]//end if Statement
]//end getNext method




]//end class
```

(B)

```
Public Class RandomLetterChooser extends RandomString[
Public RandomLetterChooser (String str)[
this.wordList =getSingleLetter(Str);

]//end constructor method

Public String[] getSingleLetters(String Str)[
String[] temp = new String [str.length];
for (int i = 0; i ≤ str.length;i++)[
 temp[i] = Str.SubString(i,i+1));
 ]//end for loop
 return temp;
 ]//end getSingleLetter method

]//end Class
```

2 A

```
Public logMessage (String message)[
for(int i = 0; i ≤ message.length; i++)[
 int temp = 0;
 if (Message.subString(i, i+1).equals(":"))[
  temp = i+ 1;
  i = message.length;
 ]//end if Statement
]//end for loop
this.MachineId = message(0, temp);
this.description = Message(temp + 1);
]//end Constructor method
```

B

```
Public boolean containsWord (String keyWord)[
boolean toF;
if(this.description.contains(keyword))[
  if(this.description.SubString(0,Keyword.length - 1).equals(keyword) || this.description.contains(" "+keyword) ||
this.description.Substring((this.description.length -1) -(keyword.length - 1),this.description.length-1).equals(keyword) ||
this.description.Contains(keyWord +" "))[
 toF = true;
 ]//end 2nd if Statement
]//end 1st if Statement
 return toF;
]//end contains word method
```

2 (c)

```
Public List<LogMessage> removeMessages(String KeyWord){
int temp = 0;
ArrayList<LogMessage> temp = new ArrayList<LogMessage>();
for(int i = 0; i < this.messageList.size()-temp;i++){
 if(this.messageList.get(i).ContainsWord(keyWord)){
  temp.add(this.messageList.get(i));
  this.messageList.remove(i);
   temp += 1;
  }//end if statement
 }//end for loop

return temp;

}//end removeMessages method
```

3) (A)

```
Private boolean toBeLabeled(int r, int c, boolean[ ]  blackSquares)[
 boolean toF = false;
if (!blackSquare [r] [c] )[
   if(blackSquare [r-1][c] || blackSquare [r][c])[
    toF = true;

  ] //end 2nd if statement

 ] //end 1st if Statement
 return toF
] //end 1st if Statement
] //end method
```

(B)

```
Public crossWord(boolean[][] blackSquares)[

this.puzzle = new Square[blackSquare.length][blackSquare[0].length]
 int temp = 1;
for(int i = 0; i < Square.length; i++)[

  for(int x=0; x<square(x).length; x++)[
   if(blackSquares[i][x])[

     puzzle[i][x] = new square(true, 0);

    ] else
   if(tobeLabled(i, x, blackSquares[i][x])[

    puzzle[i][x] = new Square(false, temp);
     temp ++;
    ] else[

    puzz[i][x] = new square(false, 0);
   ]//end enhance for loop

  ]//end for loop

  ]//end constructor method
```

4| Ⓐ

```
Public static int totalLetters(List<String> WordList)[
int temp = 0;
for(int i = 0; i < Word List.size(); i++)[
temp += WordList.get(i).length();
]//end for loop
 return temp;
]//end method
```

[Hi!], [senpai], [Boys], [girls], [Bye]

Ⓑ

```
 Public Static int basicGapWidth(List<String> wordList, int formattedLen)[

 return (formattedLen - totalLetters(wordList))/WordList.Size());
]//end method
```

Ⓒ

```
Public Static String format(List<string> WordLIST, int formatted Len)[
 int newString;
 int leftover = leftover(wordList, formatted Len);
for(int i = 0; i < WordList.Size()-1; i++)[
    newString += WordList(i);
    for(int x = 0; x < basicGapWidth(WordList,formattedLen); x++)[
     newString += " ";
   ]//end 2nd for loop
  if(leftOver > 0)[
 newString += " ";
  leftover -= 1;
  ]//end if statement
]//end for loop

return newString;

]//end method
```