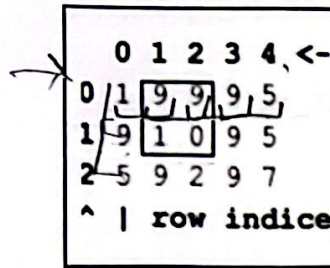
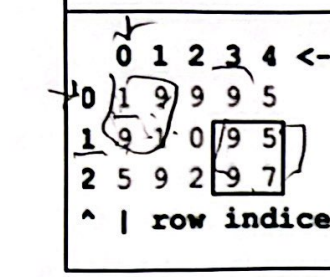
** Include the method header in your code. **

```
public static ArrayList<String> removeInfrequent(ArrayList<String> words, String inf.){
    ArrayList<String> removedWords = new ArrayList<>();
    int index=0;
    while(index < words.size()){
        String word= words.get(index).toLowerCase();
        boolean matched = true;
        int i=0;
        while(matched && i< word.length()){
            matched = inf.indexOf(word.charAt(i)) != -1;
            i++i
        }
        if(matched){
            String removed = words.remove(index);
            removedWords.add(removed);
        } else {
            index++i
        }
        return removedWords;
    }
```

	9 + 9 + 1 + 0 = 19
	9 + 5 + 9 + 7 = 30

The last example is the max sum of all the subarrays. In this case, the method would return an array equal to [1, 3].

It might be helpful to use `Integer.MIN_VALUE`, a constant representing the integer less than any other integer.

Restrictions

- Cannot use any methods from the Array class.
- You may create one array to aid in your solution.
- Do not modify the original array.

** Include the method header in your code. **

```
public static int[] getMaxSum(int[][] nums) {
    int cur_MAX = Integer.MIN_VAUE;
    int row_max = 0;
int col_Max = 0;
    for (int i=0; i <nums.length-1; i++) {
        for (int j=0; j<nums[0].length-1; j++) {
            int cur_sum = nums[i][j] +nums[i+1][j] +nums[i][j
                +nums[i+1][j+1];
            if (cur_sun > cur_Max) {
                cur_max =cur_sum;
                row_max = i;
                col_max = j;
            }
        }
    return {row_max
    col_max}
```

This is an example input file. The file contains one or more items from a single grocery bill. Each item includes a name, discount category and price. There may be more than one item per line and one item may span across multiple lines. No item names include a space. The discount category may be capitalized differently for different items. All three items will occur in the proper order for each item. The method can assume the file is structured as described.

```
avocado RED 1  blueberries none 5 milk blue
2.00
   cream red           1.00 cereal None                    1.29
```

There are three discount codes:

- red: 10% off full price
- blue: 25% off full price
- none: full price

Output

Given the input example above, the calculation would be:
`(1 * .90) + 5.0 + (2.0 * .75) + (1.0 * .9) + 1.29`

and the output would be: `9.59`

** Include the method header in your code. **

```
public static double groceryTotal(Scanner sc) {
 double red = 0.0;
 double blue = 0.0;
 double none = 0.0;
 while (sc.hasNext()) {
   scan.next();
   String keyword = scan.next().toLowerCase();
    if (keyword.equals("red")) {
       red += sc.nextDouble();
    } else if (keyword.equals("blue")) {
       blue += sc.nextDouble();
    } else {
       none += sc.nextDouble();
    }
 }
 return red * 0.9 + blue * .75 + none;
}
```

TA Use Only	
Points	Points Earned
4	

```
// constants for fighting
public static enum Attack {
    ROAR, POUNCE, SCRATCH, FORFEIT
}
```

```
public Critter.Direction getMove() {
    if (currPattern.size() > 0) {
        return currPattern.remove(0);
    } else {
        currPattern.add(NORTH);
        for (int i=0; i < eaten+1; i++)
            currPattern.add(EAST);
        currPattern.add(NORTH);
        for (int i=0; i < eaten+1; i++)
            currPattern.add(EAST);

        return currPattern.remove(0);

    }
```

```
class Dragonfly() {
    int eaten = 0;
    Arraylist<Direction> currPattern;
    int fight = 0;
    public Dragonfly() {
        currPattern = new ArrayList<>();
        currPattern.add(Direction.NORTH);
        currPattern.add(Direction.EAST);
                   .add(Direction.SOUTH);
                   .add(Direction.EAST);
    }
    public Attack fight() {
        Attack[] att = { ROAR, POUNCE, SCRATCH, FORFEIT
        Attack output = att[fight % 4];
        fight++;
        return output;
    }
    public boolean eat() {
        eaten++;
        return true;
    }
```

```
    public String getAmPm()

    // returns String for time, for example: "6:27 PM"
    public String toString()


    // your method would go here

}
```

** Include the method header in your code. **

```
Public void advance(int minutes) {
 Minute += minutes;

 while (minute >= 60) {
       minute -= 60;
       hour ++;
       if (hour == 12) {
            if (amPM.equals("AM") {
                amPM = "PM";
            } else {
                amPM = "AM";
            } else if (hour > 12) {
                hour = 1;
            }
       }
  }
```

TA Use Only						
Item	A	B	C	D	X	Totals
Possible Points	2	2	3	9	N/A	16
Points Earned						