

# CS 24 AP Computer Science A Review

## Week 10: Algorithm II

DR. ERIC CHOU  
IEEE SENIOR MEMBER



## SECTION 1

String,  
ArrayList<String>,  
char[] Conversion



# Questions

AP2016 – Q1(B)

# String, Array of Characters and ArrayList of Characters

---

```
static String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
static char[] alpha = getChar(alphabet);  
static ArrayList<String> alphaList = new ArrayList<String>();
```

- Java does not have direct conversion between these data types.
- Use character level operators effectively.

# String to array of characters

---

```
public static char[] getChars(String x){  
    char[] a = new char[x.length()];  
    for (int i=0; i<x.length(); i++) a[i]= x.charAt(i);  
    return a;  
}
```

# Array of Characters to String

```
public static String reverseToString(char[] x){  
    String str = "";  
    for (int i=x.length-1; i>=0; i--){  
        str += ""+x[i];  
    }  
    return str;  
}
```

# String to ArrayList of Strings

```
public static ArrayList<String> getList(String x){  
    ArrayList<String> a = new ArrayList<String>();  
    for (int i=0; i<x.length(); i++){  
        a.add(x.substring(i, i+1));  
    }  
    return a;  
}
```

# ArrayList of Strings to String

```
public static String reverseToString(ArrayList<String> x){
    String str = "";
    for (int i=x.size()-1; i>=0; i--){
        str += ""+x.get(i);
    }
    return str;
}
```



# Testing

```
public static void main(String[] args){
    alphaList = getList(alphabet);
    System.out.print("\f");
    System.out.println(alphabet);
    System.out.println(Arrays.toString(alpha));
    System.out.println(reverseToString(alpha));
    System.out.println(alphaList);
    System.out.println(reverseToString(alphaList));
}
```

ABCDEFGHIJKLMNOPQRSTUVWXYZ

[A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z]

ZYXWVUTSRQPONMLKJIHGFEDCBA

[A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z]

ZYXWVUTSRQPONMLKJIHGFEDCBA

SECTION 2

# Difference of a Sequence

## Common Difference

The common difference is the constant amount of change between numbers in an arithmetic sequence.

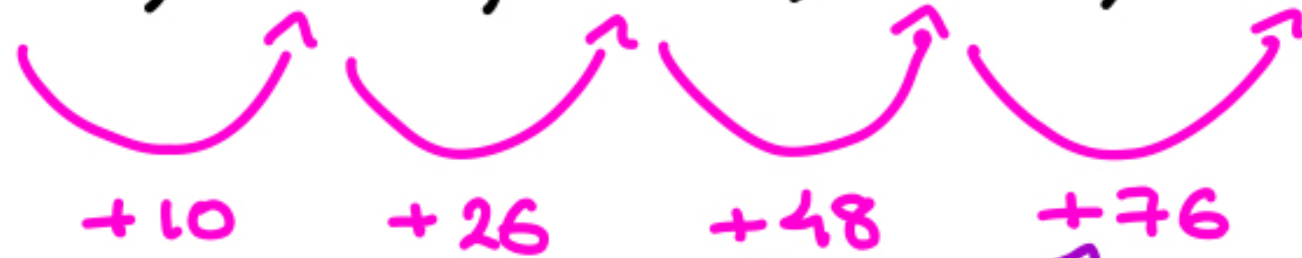
2, 4, 6, 8, ...

Curved arrows point from 2 to 4, 4 to 6, and 6 to 8. Below each arrow is a '+2'.

+2 +2 +2

common  
difference: 2

4 , 14 , 40 , 88 , 164 , ...



1<sup>st</sup> differences



2<sup>nd</sup> differences



3<sup>rd</sup> differences

## Arithmetic Sequence

A pattern of numbers that increase or decrease at a **constant amount**

2, 4, 6, 8, 10, 12 . . .



common  
difference: 2

term position

$a_n = a_1 + (n - 1)d$

$n^{\text{th}}$  term

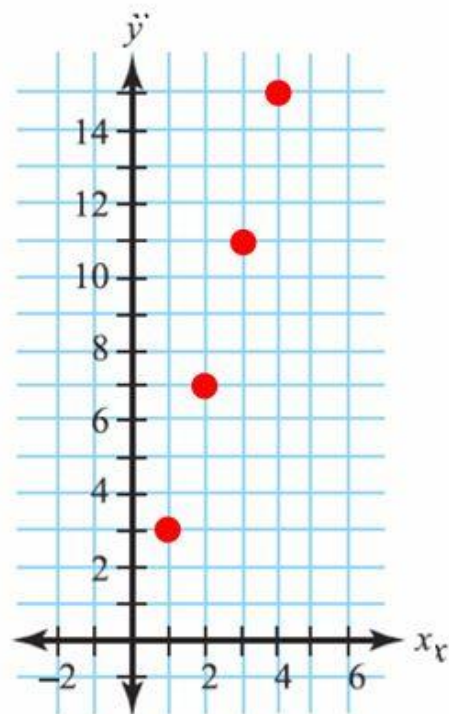
first term

common difference

The diagram shows the formula for the  $n^{\text{th}}$  term of an arithmetic sequence,  $a_n = a_1 + (n - 1)d$ , centered within a yellow rounded rectangle with a red border. Three labels with arrows point to parts of the formula: 'term position' points down to the  $n$  in the subscript of  $a_n$ ; 'first term' points up to the  $a_1$ ; and 'common difference' points up to the  $d$ . The label ' $n^{\text{th}}$  term' is positioned to the left of the formula, with an arrow pointing to the  $a_n$  term.

Complete the table, and graph the corresponding points.

	$n$	$t_n = 3 + (n - 1)4$	$t_n$
1.	1	$t_1 = 3 + (1 - 1)4$	3
2.	2	$t_2 = 3 + (2 - 1)4$	7
3.	3	$t_3 = 3 + (3 - 1)4$	11
4.	4	$t_4 = 3 + (4 - 1)4$	15



5. Write a linear function for the relationship between  $t$  and  $n$ .

$$t = 3 + 4n$$

6. What does the number 4 represent in both the graph and the sequence?

The slope and the common difference

## SITUATION 5 (money in wallet paying for sodas)

## GRAPH

John has \$ 10 in his wallet. Each soda he buys costs \$ 2.00.

### SLOPE

(rate of change) **- 2**

### Y-intercept

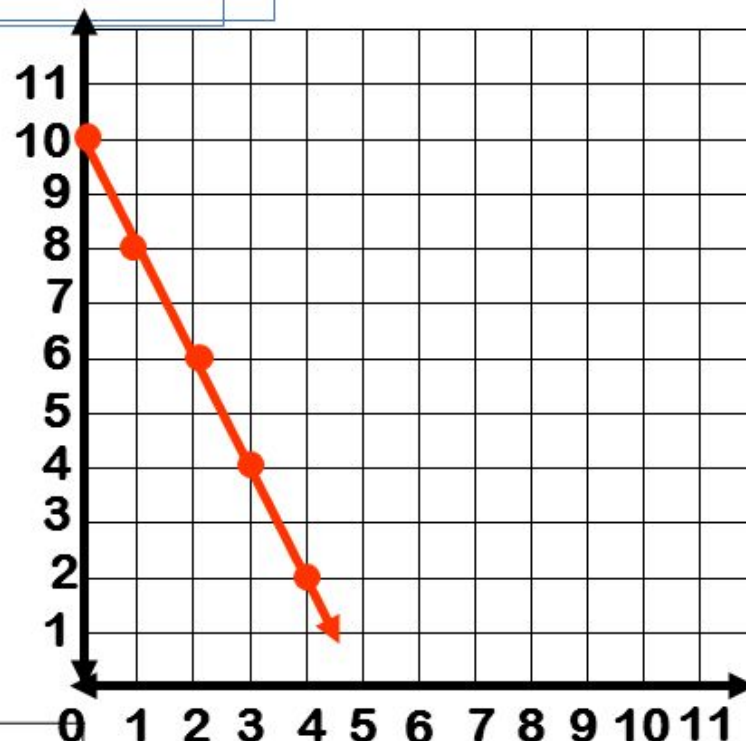
(initial value) **10**

### TABLE

Number X of sodas bought	Amount left y
0	10
1	8
2	6
3	4
4	2

### EQUATION

$$y = -2x + 10$$



### SEQUENCE

Initial Value	1st	2nd	3rd	4th
<b>10</b>	8	6	4	2

### NEXT-NOW STATEMENT

**NEXT = NOW - 2 ;  
STARTING AT 10**

Practical Domain  **$0 \leq x \leq 5$**

Practical Range  **$0 \leq y \leq 10$**



$$T_n = an^2 + bn + c$$

$$n = 1$$

$$n = 2$$

$$n = 3$$

$$n = 4$$

$$T_n$$

$$a + b + c$$

$$4a + 2b + c$$

$$9a + 3b + c$$

$$16a + 4b + c$$

1<sup>st</sup> difference

$$3a + b$$

$$5a + b$$

$$7a + b$$

2<sup>nd</sup> difference

$$2a$$

$$2a$$

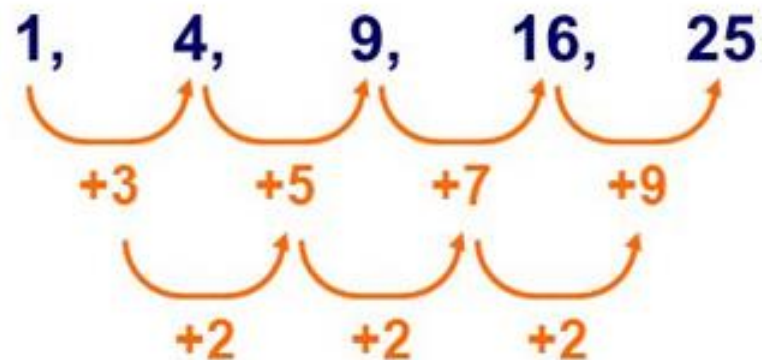
# Quadratic sequences

When the second row of differences produces a constant number the sequence is called a **quadratic sequence**. This is because the rule for the  $n^{\text{th}}$  term of the sequence is a quadratic expression of the form

$$u_n = an^2 + bn + c$$

where  $a$ ,  $b$  and  $c$  are constants and  $a \neq 0$ .

The simplest quadratic sequence is the sequence of **square numbers**.



The constant second difference is 2 and the  $n^{\text{th}}$  term is  $n^2$ .



# Questions

AP2017 – Q1(B)

## A1(a) Strictly Increasing

```
public static boolean isStrictlyIncreasing(int[] x){  
    for (int i=0; i< x.length-1; i++){  
        if (x[i+1]<=x[i]) return false;  
    }  
    return true;  
}
```

## A1(b) maximum Slope

```
public static int maxSlope(int[] x){
    int max = Integer.MIN_VALUE;
    if (x.length<2) return 0;
    for (int i=0; i< x.length-1; i++){
        if (x[i+1]-x[i]> max) { max = x[i+1]-x[i]; }
    }
    return max;
}
```

## A1(c) Minimum Slope

```
public static int minSlope(int[] x){
    int min = Integer.MAX_VALUE;
    if (x.length<2) return 0;
    for (int i=0; i< x.length-1; i++){
        if (x[i+1]-x[i]< min) { min = x[i+1]-x[i]; }
    }
    return min;
}
```

# A1(d) Maximum Delta

```
public static int maxDelta(int[] x){
    int max = Integer.MIN_VALUE;
    if (x.length<2) return 0;
    for (int i=0; i< x.length-1; i++){
        if (Math.abs(x[i+1]-x[i])> max) {
            max = (int) Math.abs(x[i+1]-x[i]);
        }
    }
    return max;
}
```

```
public static int[] a={  
    1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11  
};  
public static int[] b={  
    1, 4, 3, 5, 6, 7, 0, 9  
};  
public static int[] c={0};
```



A1 Part(a):

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] is strictly increasing = true

[1, 4, 3, 5, 6, 7, 0, 9] is strictly increasing = false

[0] is strictly increasing = true

A1 Part(b):

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] maxSlope = 1

[1, 4, 3, 5, 6, 7, 0, 9] maxSlope = 9

[0] maxSlope = 0

A1 Part(c):

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] minSlope = 1

[1, 4, 3, 5, 6, 7, 0, 9] minSlope = -7

[0] minSlope = 0

A1 Part(d):

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] maxDelta = 1

[1, 4, 3, 5, 6, 7, 0, 9] maxDelta = 9

[0] maxDelta = 0

# Key Points

---

- Boundary condition: null pointer, empty array (length==0), single element array.

## SECTION 3

# Find() function



# Questions

AP2017– Q3(A, B, C)

# Find a pattern in a string

---

- find: Find the first occurrence.
- findN: Find the Nth occurrence without overlapping patterns.
- findL: Find the last occurrence without overlapping patterns.
- findAN: Find the Nth occurrence with overlapping patterns.
- findAL: Find the Last occurrence with overlapping patterns.

# Find

This one is the same as indexOf

```
public static int find(String str, String pattern){
    if (str.indexOf(pattern) >=0) return str.indexOf(pattern);
    return -1;
}
```

# findN

```
public static int findN(String str, String pattern, int n){
    int i=-1;
    int pos =0;
    int count = 0;
    while (str.indexOf(pattern, pos) >=0 && count != n) {
        i = str.indexOf(pattern, pos);
        pos = i+1;
        count++;
    }
    if (count != n) return -1;
    return i;
}
```

# findL

```
public static int findL(String str, String pattern){
    int i = -1;
    int pos = 0;
    while (str.indexOf(pattern, pos) >= 0) {
        i = str.indexOf(pattern, pos);
        pos = i+1;
    }
    return i;
}
```



# findAN

```
public static int findAN(String str, String pattern, int n){
    int i=-1;
    int pos =0;
    int count = 0;
    while (str.indexOf(pattern, pos) >=0 && count != n) {
        i = str.indexOf(pattern, pos);
        pos = i+pattern.length();
        count++;
    }
    if (count != n) return -1;
    return i;
}
```

# findAL

```
public static int findAL(String str, String pattern){
    int i = -1;
    int pos = 0;
    while (str.indexOf(pattern, pos) >= 0) {
        i = str.indexOf(pattern, pos);
        pos = i + pattern.length();
    }
    return i;
}
```

```

public static void main(){
    System.out.print("\f");
    System.out.println("A2 Part(a):");
    System.out.println("0      1      2      3      4      5      6      7");
    System.out.println("012345678901234567890123456789012345678901234567890123456789");
    System.out.println(a);
    System.out.println(find(a.toLowerCase(), "ha"));
    System.out.println(findN(a.toLowerCase(), "ha", 6)); // Overlapping
    System.out.println(findL(a.toLowerCase(), "ha"));
    System.out.println(findAN(a.toLowerCase(), "ha", 6)); // Non-Overlapping
    System.out.println(findAL(a.toLowerCase(), "ha"));
    System.out.println("A2 Part(b):");
    String b = "bbb bbb bbbbbb";
    System.out.println(b);
    System.out.println(find(b.toLowerCase(), "bb"));
    System.out.println(findN(b.toLowerCase(), "bb", 3)); // Overlapping
    System.out.println(findL(b.toLowerCase(), "bb"));
    System.out.println(findAN(b.toLowerCase(), "bb", 3)); // Non-Overlapping
    System.out.println(findAL(b.toLowerCase(), "bb"));
}

```

A2 Part(a):

0	1	2	3	4	5	6	7
0123456789012345678901234567890123456789012345678901234567890123456789							
Ha Ha Ha Ha! HaHaHa! I am a happy person, HaHaHa! I have a million dollar.							

0

15

52

15

52

A2 Part(b):

bbb bbb bbbbbb

0

4

11

8

10

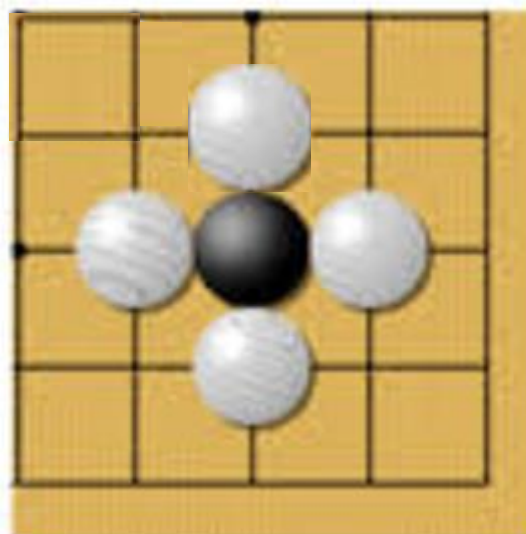
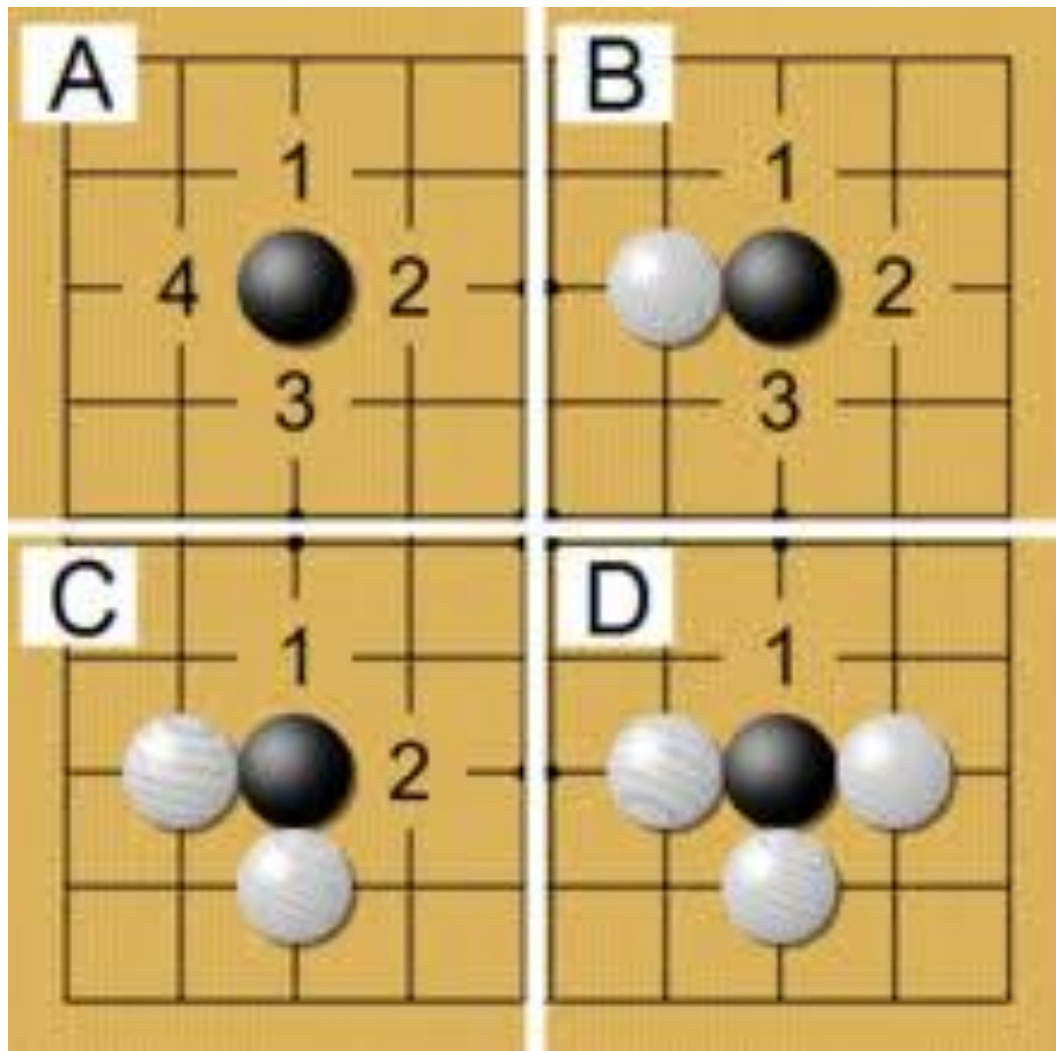
SECTION 4

# 2-D Traversal Dead Go-Game Stone Detection



# Questions

AP2016– Q3(A, B, C)



### Stone Colors:

String B="B";

String W="W";

String S=" ";

### Single Dead Stone:

1. A stone is dead if it is surrounded by stones of other color.
2. Stones on a side is dead if surrounded by 3 sides.
3. Stones at a corner is dead if surrounded by 2 sides.

We don't detect the death of connected stones.

# Generate a Scanned Map for the Dead Stones

---

- Game Board Size 9x9.
- `public static boolean isDead(String[][] gameboard, int r, int c);`
- Input Parameter: a 9x9 Go-Game game board.
- Return: if a stone is dead. (  $0 \leq r < 9$ ,  $0 \leq c < 9$  )



```

1 import java.util.Scanner;
2 public class Go{
3     public static String B = "B";
4     public static String W = "W";
5     public static String S = " ";
6     public static String[][] gameboard ={
7         {S, S, S, S, W, B, W, S, S},
8         {S, S, S, S, B, W, S, S, S},
9         {S, S, S, S, W, S, S, S, S},
10        {S, B, B, S, S, S, S, S, S},
11        {S, B, W, B, S, S, S, S, S},
12        {W, W, B, S, S, S, S, W, S},
13        {W, B, S, S, S, S, W, B, W},
14        {B, W, S, S, S, S, B, W, B},
15        {W, S, S, S, S, S, S, B, W}
16    };
17    public static void display(String[][] m){
18        System.out.println("  0 1 2 3 4 5 6 7 8");
19        for (int i=0; i<m.length; i++){
20            System.out.print(i+" ");
21            for (int j=0; j<m[i].length; j++){
22                System.out.print(m[i][j]+" ");
23            }
24            System.out.println();
25        }
26    }

```

	0	1	2	3	4	5	6	7	8
0					W	B	W		
1					B	W			
2					W				
3		B	B						
4		B	W	B					
5	W	W	B					W	
6	W	B					W	B	W
7	B	W					B	W	B
8	W							B	W

Enter row number (999 to exit):6

Enter column number:0

Stone[6,0]=false

Enter row number (999 to exit):999

## isDead() logic

---

- The stone itself is not S.
- The stone's any neighbor is not S, and not the same as the stone.
- If a side is board boundary, that side is considered "Surrounded" status.

# isDead

```
public static boolean isDead(String[][] board, int r, int c){
    boolean top=false, left=false, bottom=false, right=false;
    String stoneColor = board[r][c];
    boolean isSpace = board[r][c].equals(S);
    if (r==0) top = true;
    else top = !isSpace && !board[r-1][c].equals(S) && !stoneColor.equals(board[r-1][c]);
    if (c==0) left = true;
    else left = !isSpace && !board[r][c-1].equals(S) && !stoneColor.equals(board[r][c-1]);
    if (r==board.length-1) bottom = true;
    else bottom = !isSpace && !board[r+1][c].equals(S) && !stoneColor.equals(board[r+1][c]);
    if (c==board.length-1) right = true;
    else right = !isSpace && !board[r][c+1].equals(S) && !stoneColor.equals(board[r][c+1]);
    return top && bottom && left && right;
}
```



# Demo Program: Go.java

GO BLUEJ!!!

```

1 public class TestGo
2 {
3     public static String B = Go.B;
4     public static String W = Go.W;
5     public static String S = Go.S;
6     public static String[][] gameboard = Go.gameboard;
7     public static void display(String[][] m){
8         System.out.println(" 0 1 2 3 4 5 6 7 8");
9         for (int i=0; i<m.length; i++){
10             System.out.print(i+" ");
11             for (int j=0; j<m[i].length; j++){
12                 System.out.print(m[i][j]+" ");
13             }
14             System.out.println();
15         }
16     }
17     public static void display(int[][] m){
18         System.out.println(" 0 1 2 3 4 5 6 7 8");
19         for (int i=0; i<m.length; i++){
20             System.out.print(i+" ");
21             for (int j=0; j<m[i].length; j++){
22                 System.out.print(m[i][j]+" ");
23             }
24             System.out.println();
25         }
26     }

```

```

17 public static void display(int[][] m){
18     System.out.println(" 0 1 2 3 4 5 6 7 8");
19     for (int i=0; i<m.length; i++){
20         System.out.print(i+" ");
21         for (int j=0; j<m[i].length; j++){
22             System.out.print(m[i][j]+" ");
23         }
24         System.out.println();
25     }
26 }
27 public static int[][] getDeadMap(String[][] m){
28     int[][] n= new int[m.length][m.length];
29     for (int i=0; i<m.length; i++){
30         for (int j=0; j<m.length; j++){
31             if (Go.isDead(m, i, j)) n[i][j] = 1;
32             else n[i][j]=0;
33         }
34     }
35     return n;
36 }

```

```

public static void main(String[] args){
    System.out.println("\fGame Board:");
    display(gameboard);
    System.out.println();
    System.out.println("Dead Map:");
    int[][] n = getDeadMap(gameboard);
    display(n);
}

```

Game Board:

	0	1	2	3	4	5	6	7	8
0					W	B	W		
1					B	W			
2					W				
3		B	B						
4		B	W	B					
5	W	W	B				W		
6	W	B					W	B	W
7	B	W					B	W	B
8	W							B	W

Dead Map:

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0
7	1	0	0	0	0	0	0	1	1
8	0	0	0	0	0	0	0	0	1

# Key Points

---

- Use Boolean variable to simplify the Boolean expression.
- Setting boundary condition wisely. Be careful about index out of bound exceptions.

SECTION 5

# Reader Head (Iterator)





# Questions

AP2017 – Q2



# Class ReaderHead

---

- Build a ReaderHead class
- Data field: **max** (number of reading addresses)  
**right** (move right?)  
**index** (current reading address)
- Methods: `getIndex()`, `isFacingRight()`,  
`changeDirection()`, `reset()`, `next()`, `move(n)`

# Methods

---

- **getIndex():** get the current reading head location
- **isFacingRight():** check if the reading head is facing right.
- **changeDirection():** change the direction of the reading head.
- **reset():** reset to the index==0 and facing right.
- **next():** move one location (right or left).
- **move(n):** move n location (right or left but will stop at ends)

```

1 public class ReaderHead{
2     private int max = 0;
3     private int index = 0;
4     private boolean right = true;
5     ReaderHead(){}
6     ReaderHead(int max){
7         this.max = max;
8         index = 0;
9         right = true;
10    }
11    public int getIndex(){ return index; }
12    public void changeDirection(){
13        right = !right;
14    }
15    public boolean isFacingRight(){
16        return right;
17    }
18    public void move(int n){
19        if (isFacingRight()){
20            while (index < max-1 && n > 0) { index++; n--; }
21        }
22        else{
23            while (index > 0 && n > 0) { index--; n--; }
24        }
25    }

```

```

31 public void next(){
32     if (isFacingRight() && getIndex() == max-1){
33         changeDirection();
34         move(1);
35         return;
36     }
37     if (!isFacingRight() && getIndex() == 0){
38         changeDirection();
39         move(1);
40         return;
41     }
42     move(1);
43 }

```

# Class TestReaderHead

---

- Using both next(), reset() and move(n) to control the reader head.

```

2 public class TestReaderHead
3 {
4     public static int[] data = {10, 20, 30, 40, 50};
5     public static void main(String[] args){
6         ReaderHead r = new ReaderHead(data.length);
7         System.out.print("\f");
8         System.out.println("B1 Part(a):");
9         for (int i=0; i<data.length*3; i++){
10             System.out.println("Index "+r.getIndex()+" : "+data[r.getIndex()]);
11             r.next();
12             if (i%data.length==data.length-1) System.out.println();
13         }
14         System.out.println("B1 Part(b):");
15         r.reset();
16         r.move(3);
17         System.out.println("Index "+r.getIndex()+" : "+data[r.getIndex()]);
18         r.move(3);
19         System.out.println("Index "+r.getIndex()+" : "+data[r.getIndex()]);
20         r.changeDirection();
21         r.move(3);
22         System.out.println("Index "+r.getIndex()+" : "+data[r.getIndex()]);
23         r.move(3);
24         System.out.println("Index "+r.getIndex()+" : "+data[r.getIndex()]);
25     }
26 }

```

B1 Part(a):

Index 0: 10

Index 1: 20

Index 2: 30

Index 3: 40

Index 4: 50

Index 3: 40

Index 2: 30

Index 1: 20

Index 0: 10

Index 1: 20

Index 2: 30

Index 3: 40

Index 4: 50

Index 3: 40

Index 2: 30

B1 Part(b):

Index 3: 40

Index 4: 50

Index 1: 20

Index 0: 10



# Iterator()

---

- Serialize the access of a data structure.
- Access sequential access devices.

SECTION 6

# Available List



# Questions

AP2016 – Q1

# Selection Sort

---

- Selection Sort using available list.
- Mark the number out if a number is picked.

```

public static ArrayList<Integer> selectionSort(int[] data){
    ArrayList<Integer> a = new ArrayList<Integer>();
    boolean[] b = new boolean[data.length];
    for (int i=0; i<b.length; i++) b[i] = true;
    for (int i=0; i<data.length; i++){
        int min = Integer.MAX_VALUE;
        int index = -1;
        for (int j=0; j<data.length; j++){
            if (b[j]){
                if (data[j] < min)
                { min = data[j];
                  index = j;
                }
            }
        }
        b[index] = false;
        a.add(min);
    }
    return a;
}

```

BlueJ: Terminal Window - Week10

Options

[0, 1, 2, 3, 5, 6, 7, 8, 9]

# Hotel class

---

## Build a Hotel Class

### Data fields:

- num: number of rooms;
- available: number of available rooms;
- alist[]: available list;

### Methods:

- isAvailable(int n): check if a room is available or not.
- isFull(): check if the hotel is full;
- checkIn(int n): check in a room if it is available, make it unavailable. And decrease the number of available rooms.
- checkOut(int n): check out a room if it is not available, make it available. And increase the number of available rooms.

```

1 public class Hotel{
2     private int num;
3     private int available;
4     private boolean[] alist;
5     Hotel(int num){
6         this.num = num;
7         available = num;
8         alist = new boolean[num];
9         for (int i=0; i<alist.length; i++) alist[i]=true;
10    }
11    public boolean isAvailable(int n){
12        if (n<num && alist[n]) return true;
13        return false;
14    }
15    public boolean isFull(){ return available <=0; }
16    public void checkIn(int n){
17        if (isAvailable(n)) {
18            alist[n] = false;
19            available--;
20        }
21    }

```

```

22    public void checkOut(int n){
23        if (!isAvailable(n)) {
24            alist[n] = true;
25            available++;
26        }
27    }
28    public String toString(){
29        String str = "[";
30        for (int i=0; i<alist.length; i++){
31            if (i==0) if (alist[i]) str += "T"; else str += "F";
32            if (i!=0) if (alist[i]) str += " T"; else str += " F";
33            if (i== alist.length-1) str += "]";
34            if (i%10== 9 || i== alist.length-1) str += "\n";
35        }
36        return str;
37    }
38 }

```

```

1
2 public class TestHotel
3 {
4     public static void main(String[] args){
5         Hotel h = new Hotel(20);
6         System.out.print("\f");
7         System.out.println("Initialized");
8         System.out.print(h);
9         h.checkIn(3);
10        h.checkIn(6);
11        h.checkIn(7);
12        System.out.println("After check-ins");
13        System.out.print(h);
14        h.checkOut(15);
15        h.checkOut(6);
16        h.checkOut(3);
17        System.out.println("After check-outs");
18        System.out.print(h);
19    }
20 }

```

```

Initialized
[T T T T T T T T T T
 T T T T T T T T T T]
After check-ins
[T T T F T T F F T T
 T T T T T T T T T T]
After check-outs
[T T T T T T T F T T
 T T T T T T T T T T]

```



SECTION 7

# Non-Recurring Set and Occurrence List

# Generation of Non-Recurring Set

---

- Add a word into a word list if the word is not contained by the word list.
- Use the occurrence list to compare and generate the occurrence list.

# Non-Recurring Set

```
ArrayList<String> wlist = new ArrayList<String>();
String[] tokens = str.split("");
for (int i=0; i<tokens.length; i++) {
    tokens[i] = tokens[i].trim();
    if (tokens[i].length() !=0 && !wlist.contains(tokens[i])) wlist.add(tokens[i]);
}
```

You may try to write your own contains() function.

# Generation of Occurrence Array (or List)

```
int[] wCount = new int[wlist.size()];
for (int i=0; i<tokens.length; i++){
    for (int j=0; j<wlist.size(); j++){
        if (wlist.get(j).equals(tokens[i])) wCount[j]++;
    }
}
```

```
1 import java.util.ArrayList;
2 public class WordList
3 {
4     static String str = "A B C A B C D E F F A B C W S Z K Z";
5
6     public static void main(String[] args){
7         ArrayList<String> wlist = new ArrayList<String>();
8         String[] tokens = str.split(" ");
9         for (int i=0; i<tokens.length; i++) {
10             tokens[i] = tokens[i].trim();
11             if (tokens[i].length() !=0 && !wlist.contains(tokens[i])) wlist.add(tokens[i]);
12         }
13         int[] wCount = new int[wlist.size()];
14         for (int i=0; i<tokens.length; i++){
15             for (int j=0; j<wlist.size(); j++){
16                 if (wlist.get(j).equals(tokens[i])) wCount[j]++;
17             }
18         }
19         System.out.print("\f");
20         System.out.println("Original String="+str);
21         for (int i=0; i<wlist.size(); i++){
22             System.out.println(String.format("Word(%d)=", i)+wlist.get(i)+" happens "+wCount[i]+" times.");
23         }
24     }
25 }
```

Original String=A B C A B C D E F F A B C W S Z K Z

Non-recurring Set=[A, B, C, D, E, F, W, S, Z, K]

Word(0)=A happens 3 times.

Word(1)=B happens 3 times.

Word(2)=C happens 3 times.

Word(3)=D happens 1 times.

Word(4)=E happens 1 times.

Word(5)=F happens 2 times.

Word(6)=W happens 1 times.

Word(7)=S happens 1 times.

Word(8)=Z happens 2 times.

Word(9)=K happens 1 times.

## SECTION 8

# Selected List



# Questions

AP 2016-Q2(C)





# Failed Students

---

- Remove students who has GPA lower than 2.0 from a school student list and put them into a list for summer class.

```
1 public class Student
2 {
3     private double GPA;
4     private String name;
5     Student(String n, double g){
6         name = n;
7         GPA = g;
8     }
9     public double getGPA(){ return GPA; }
10    public String getName(){ return name; }
11    public String toString(){return "{"+name+", "+GPA+"}"; };
12 }
```


```

1 import java.util.ArrayList;
2 import java.util.Arrays;
3 public class WashingtonHigh
4 {
5     public static ArrayList<Student> getSummerClass(ArrayList<Student> alist){
6         ArrayList<Student> summer = new ArrayList<Student>();
7         for (int i=alist.size()-1; i>=0; i--){
8             if (alist.get(i).getGPA() < 2.0){
9                 summer.add(alist.remove(i));
10            }
11        }
12        return summer;
13    }
14    public static void main(String[] args){
15        ArrayList<Student> slist = new ArrayList<Student>(
16            Arrays.asList( new Student[]{
17                new Student("Amy", 2.85), new Student("Bryan", 3.6), new Student("Carol", 1.46),
18                new Student("David", 4.0), new Student("Ellie", 1.98), new Student("Fanny", 3.2),
19                new Student("Goerge", 1.65), new Student("Harry", 2.78), new Student("Ivan", 2.95),
20                new Student("Jack", 1.8), new Student("Kate", 3.3), new Student("Larry", 2.25),
21                new Student("Mary", 2.03), new Student("Nancy", 1.77), new Student("Omar", 3.6),
22                new Student("Peter", 2.66), new Student("Queen", 1.99), new Student("Robert", 3.98)
23            })
24        );
25        System.out.println(getSummerClass(slist));
26    }
27 }

```

# Summer Class Listing

---

 BlueJ: Terminal Window - Week9

Options

```
[ {Queen,1.99}, {Nancy,1.77}, {Jack,1.8}, {Goerge,1.65}, {Ellie,1.98}, {Carol,1.46}]
```

SECTION 9

# Custom Design String Split Function

# How to split string into tokens?

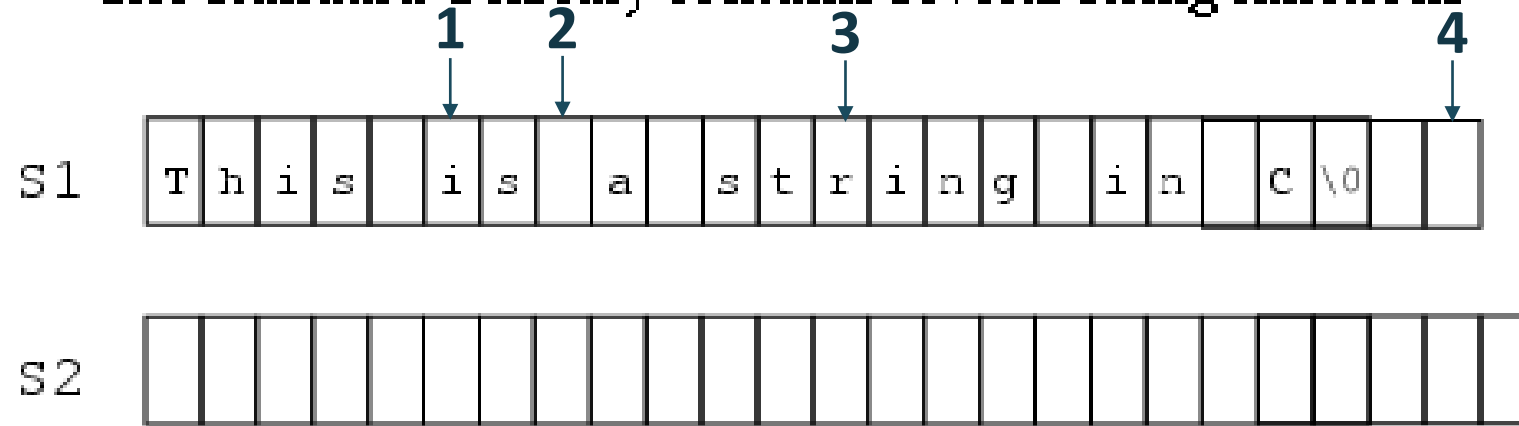
---

Detect the start of a token and the end of a string.

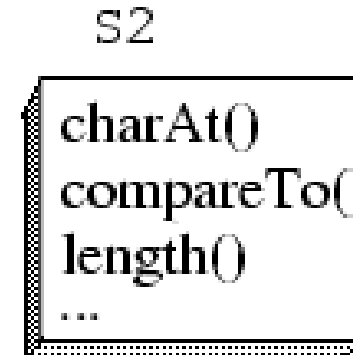
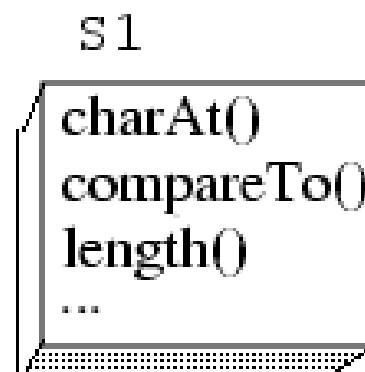
1. isSpace and the next character is not space -> start of a token
2. !isSpace and the next character is space -> end of a token
3. !isSpace and the next character is not a space -> in the middle of a token
4. isSpace and the next character is also a space -> do nothing and it is in space sequence.

A String in C is a nul-terminated sequence of chars

The standard C library contains several string functions



A String in Java is an object with several methods



```
13 public String[] split(){
14     ArrayList<String> a = new ArrayList<String>();
15     if (str.length()==0) return null;
16     if (str.length()==1) return new String[]{str};
17     str = str.trim();
18     int i=0;
19     boolean isSpace = true;
20     String words = "";
21     while (i<str.length()){
22         if (isSpace && !str.substring(i, i+1).equals(" ")) {
23             words+=str.substring(i, i+1);
24             isSpace = !isSpace;
25         }
26         else if (!isSpace && !str.substring(i,i+1).equals(" ")){
27             words+=str.substring(i, i+1);
28         }
29         else if (!isSpace && str.substring(i,i+1).equals(" ")){
30             a.add(words);
31             words="";
32             isSpace = !isSpace;
33         }
34         i++;
35         if (i==str.length() && words.length() !=0) a.add(words);
36     }
37     String[] b = new String[a.size()];
38     for (int k=0; k<a.size(); k++) b[k]=a.get(k);
39     return b;
40 }
```



```

1 import java.util.ArrayList;
2 import java.util.Arrays;
3 public class MySplit
4 {
5     private static String str1 = "    AAA  BB  CC D  EEEEEEE FFFF ";
6     private static String str2 = "AAA  BB  CC D  EEEEEEE";
7     private static String str3 = "AAA  BB  CC D  EEEEEEE ";
8     private static String str4 = "    AAA  BB  CC D  EEEEEEE";
9     private static String str5 = "";
10    private static String str6 = "A";
11    private String str="";
12    MySplit(String s){str = s; }

```

```

42 public static void main(String[] args){
43     MySplit m1 = new MySplit(str1);
44     System.out.println(Arrays.toString(m1.split()));
45     MySplit m2 = new MySplit(str2);
46     System.out.println(Arrays.toString(m2.split()));
47     MySplit m3 = new MySplit(str3);
48     System.out.println(Arrays.toString(m3.split()));
49     MySplit m4 = new MySplit(str4);
50     System.out.println(Arrays.toString(m4.split()));
51     MySplit m5 = new MySplit(str5);
52     System.out.println(Arrays.toString(m5.split()));
53     MySplit m6 = new MySplit(str6);
54     System.out.println(Arrays.toString(m6.split()));
55 }

```

## Options

```

[AAA, BB, CC, D, EEEEEEE, FFFF]
[AAA, BB, CC, D, EEEEEEE]
[AAA, BB, CC, D, EEEEEEE]
[AAA, BB, CC, D, EEEEEEE]
null
[A]

```

## SECTION 10

# Toggler

# Binary Toggler

---

- Initial Condition
- BinaryToggler Kernel Function:  
 $t = !t;$
- BinaryToggler Transformation  

```
BinaryToggler t = new BinaryToggler(true);  
t.next();
```

1	public class BinaryToggler	
2	{	true
3	boolean t=false;	false
4	BinaryToggler(boolean x){t = x;}	
5	public void next(){	true
6	t = !t;	false
7	}	
8	public boolean get(){	true
9	return t;	false
10	}	
11	public String toString(){ return ""+t;}	true
12		
13	public static void main(String[] args){	false
14	BinaryToggler bt = new BinaryToggler(true);	true
15	for (int i=0; i<10; i++){	
16	System.out.println(bt);	false
17	bt.next();	
18	}	
19	}	
20	}	

# Toggler Kernel

---

- Initial state of the toggler = -1
- Toggler Kernel

```
state = (state+1) % count; // count is number of states
```
- Toggler Transformation

```
Toggle t = new Toggle(count);  
t.toggle()*step+base; // use toggle function like random()
```

```

8 public class Toggler
9 {
10     int state = -1;
11     int count = 2;
12     Toggler(int count){ this.count = count; }
13
14     public int toggle(){ state = (state+1) % count; return state; }
15
16
17     public static void main(String[] args){
18         System.out.println("\fPart (a) 3-way Toggle:");
19         Toggler t = new Toggler(3);
20         for (int i=0; i<10; i++){
21             System.out.println(t.toggle());
22         }
23         int count = 4;
24         int step = 2;
25         int base = 10;
26         System.out.println("\fPart(b) 4-way Toggle: 10, 12, 14, 16 Toggle.");
27         Toggler t1 = new Toggler(count);
28         for (int i=0; i<10; i++){
29             System.out.println((t1.toggle()*step+base));
30         }
31     }
32 }

```

Part (a) 3-way Toggle:

0

1

2

0

1

2

0

1

2

0

Part(b) 4-way Toggle:

10

12

14

16

10

12

14

16

10

12