Homework

Palindrome check

“racecar” “madam” “abca” “saasas”

- “a” “”

“ now is i won “ “ “

Palindrome Algorithm

```
Left - leftmost character                          <<skip spaces
Right - rightmost character                        <<skip spaces

while left < right…
    if  charAt(left) != charAt(right)
        return FALSE - not a palindrome

    move left, move right                          <<skip spaces

return TRUE - is a palindrome
```

Arrays, Stacks and Queues

Arrays (chapter 8.1 & 8.2)

Linear (ordered) list of items

Java syntax

declaring an array

```
int [ ] arr = new int [MY_MAX_VALUES];
int [ ] counters = new int[NUM];
```

access

```
arr[j] = 872;       or    x = arr[i];

while (nums[i] < END_VALUE) …
```

Uses

Many, many; Usually associated with loops

Arrays

Assume you have a list of *N* integers

Find the maximum

Find the minimum

Find the mean (average)

Arrays of ***Objects***

Different than “arrays of fundamental types”
ints, float, char, boolean

```
House [ ] development;

House [ ] development = new House[NUM_HOUSE];

for (int i = 0; i < NUM_HOUSE; i++)
   development[i] = new House( );
```

Array Implementation

Fixed length collection of items

Indexed 0 - (n-1)

Array items must be same (basic) class / type

  Homogenius

*Can use inheritance or interfaces to allow

  Heterogenius

Stack

Stack is a collection of objects in which the order of access is defined by Last-In, First-Out (LIFO) policy

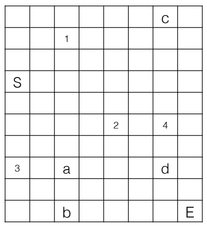
Access to items on the stack is limited to the "top" item. Other items are not available.

API (basic)

```
boolean isEmpty ( )
push (item)   //put item onto the stack
pop ( )       //returns & removes value on top
peek ( )      //optional - only looks at the top item,
                 does not remove
```

Applications

```
Backtracking - Maze exploration
  could go N S E W
  Start at 'S'
  while not at 'E'
    go till "choices" or dead end
    if choices
        push the choices avail (remember them)
    if (stack not empty) pop stack to current loc
```



Applications

"Call Stack" / "Program Stack"

Keeps track of calling/returning from methods

"Calling" a method PUSHes address of next instruction on the stack

"Return" from a method POPs address into PC and execution continues

Applications

```
main {
  soo (5)
  bar (9)
}

soo (int x) {
   bar(x+1)
}

bar (int y) {
   print "Hello:"+y;
}
```

Computer programs
- keep track of next instruction
- Program Counter
  - starts at 1001 (first line of main)
  - call soo() - push (PC+1) on stack
    - set PC to address of soo's first instruction
    - Execute (at PC = 1005)
    - call bar() - push (PC+1) on stack
      - set PC to address of bar's first instruction (1008)
      - Execute instruction until 1009 (end of method)
      - **Pop** value off stack into —> PC
    - Execute at (PC) …

Parenthesis matching

Push on left

Pop on right

Balanced if empty at end!

There are easier ways…

Queue

First in - First Out ordered list

“Queue” up for the movies; or a bank teller;

Queue

Queue is a collection of objects in which the order of access is defined by First-In, First-Out (FIFO) policy

Add to items to the “end” of the queue; Remove items from the “front” of the queue. Other items are not available.

```
API  (basic)
   int count ( )
   enqueue (item)   //put item onto the queue at end
   dequeue ( )        //returns & removes value from front
   peek ( )        //only looks at the front item,  does not remove
```

Applications of Queues

Keeping list of things that need

First-come, first-served priority

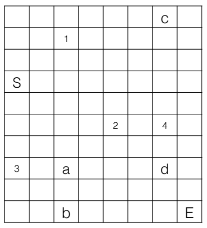
Buffer between systems of different speeds

one system enqueues; other dequeues

Backtracking - but do it “Breadth-first”

Applications (Queue)

```
Backtracking -  Maze exploration
   could go N S E W
   Start at ’S’
   while not at ‘E’
      go till “choices” or dead end
      if “choices”
            enqueue all choices avail (remember them)
      if (queue not empty) dequeue as  current loc
```


c
1
S
2
4
3
a
d
b
E