Pseudocode for IB CS examinations - Hockerill guide

This guide is a summary of pseudo code taken from three IBO documents:

1. The computer science guide (First examinations 2014)
2. Approved notation for developing pseudocode
3. Pseudocode in Examinations

Conventions	Variable names are all capitals, for example, CITY. Pseudocode keywords are lowercase, for example, loop, if … Method names are mixed case, for example, getRecord(). Methods of an object are invoked using the “dot notation” used in most languages, for example, BIGARRAY.binarySearch(27)

Variable names	These will be provided and comments // used, for example:
	N = 5 // the number of items in the array
	SCOREHISTORY,getExam(NUM) // get the student’s score on exam NUM

Assignment	Values will be assigned using = , for example: N = 5 // indicates the array has 5 data items VALUE[0] = 7 // assigns the first data item in the array a value of 7
Output of information	Output—this term is sufficient to indicate the data is output to a printer, screen, for example: output COUNT // display the count on the screen
Arrays	An array is an indexed and ordered set of elements. Unless specifically defined in the question, the index of the first element in an array is 0. NAMES[0] // The first element in the array NAMES
Strings	A string can contain a set of characters, or can be empty. Strings can be used like any other variable. MYWORD = "This is a string" if MYWORD = "the" then output MYWORD end if Strings should be regarded as a class of objects. If a specialized method such as charAt() or substring() is to be used in an examination it will be fully specified as part of the question. This explanation might be given *“charAt(x) returns the character found at position x or NIL if x is out of the bounds of the string* *MYWORD = “pseudocode”* *Output(MYWORD.charAt(4) // The character ‘d’ would be outputted”*
Collections	Collections store a set of elements. The elements may be of any type (numbers, objects, arrays, Strings, etc.). A collection provides a mechanism to iterate through all of the elements that it contains (See examples below)

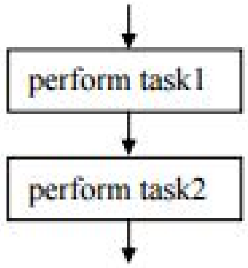
Symbol	Definition	Examples	
=	is equal to	X = 4, X = K	If X = 4
>	is greater than	X > 4	if X > 4 then
>=	is greater than or equal to	X >= 6	loop while X >= 6
<	is less than	VALUE[Y] < 7	loop until VALUE[Y] < 7
<=	is less than or equal to	VALUE[] <=12	if VALUE[Y] <= 12 then
≠	not equal to	X ≠ 4, X ≠ K	
AND	logical AND	A AND B	if X < 7 AND Y > 2 then
OR	logical OR	A OR B	if X < 7 OR Y > 2 then
NOT	logical NOT	NOT A	if NOT X = 7 then
mod	modulo	15 mod 7 = 1	if VALUE[Y] mod 7 = 0 then
div	integer part of quotient	15 div 7 = 2	if VALUE[Y] div 7 = 2 then

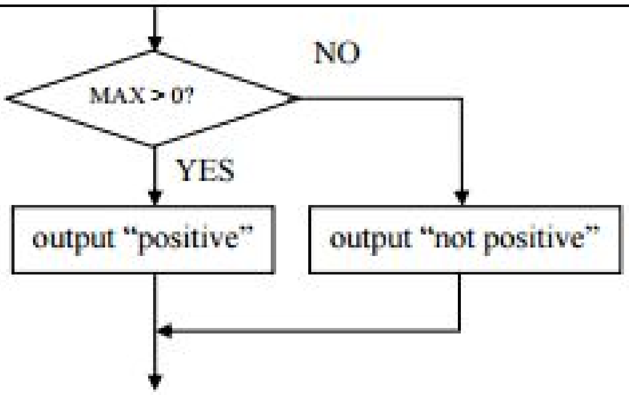
FACTORS

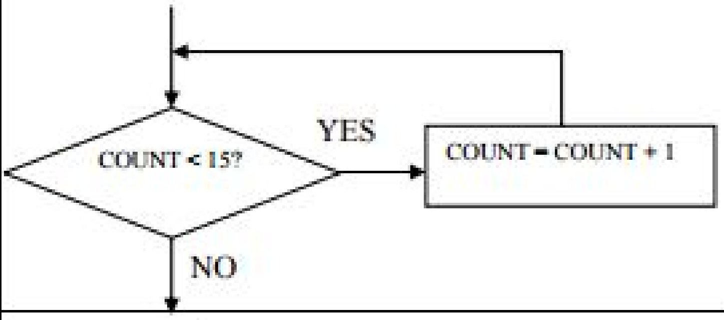
The following pseudocode presents an algorithm that will print all the factors of an integer. It prints two factors at a time, stopping at the square root. It also counts and displays the total number of factors.

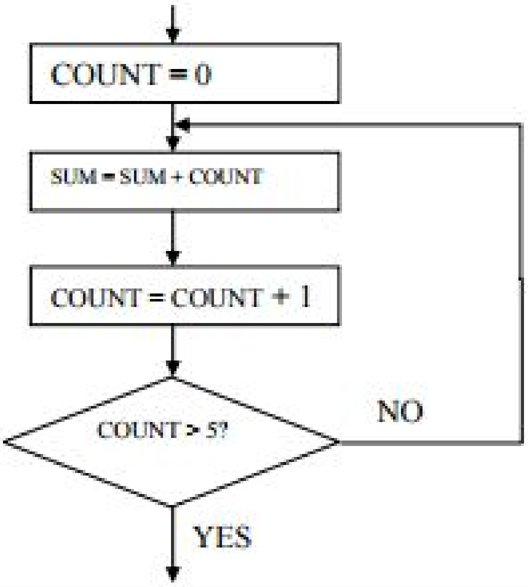
```
// recall that
//   30 div 7 = 4
//   30 mod 7 = 2

NUM = 140  // code will print all factors of this number
F = 1
FACTORS = 0

loop until F*F > NUM  //code will loop until F*F is greater than NUM

  if NUM mod F = 0 then

    D = NUM div F
    output NUM , " = " , F , "*" , D

    if F = 1 then
      FACTORS = FACTORS + 0
    else if F = D then
      FACTORS = FACTORS + 1
    else
      FACTORS = FACTORS + 2
    end if

  end if

  F = F + 1

end loop

output NUM , " has " , FACTORS , " factors "
```

FLOWCHARTS

Operation	Flowchart example	Pseudocode example
sequential operations		perform task1 perform task2

conditional operations	 MAX > 0?
	NO
	YES
	output "positive"
	output "not positive"
		if MAX > 0 then
		output "positive"
		else
		output "not positive"
		end if

while-loop		loop while COUNT < 15
		COUNT = COUNT + 1
		end loop

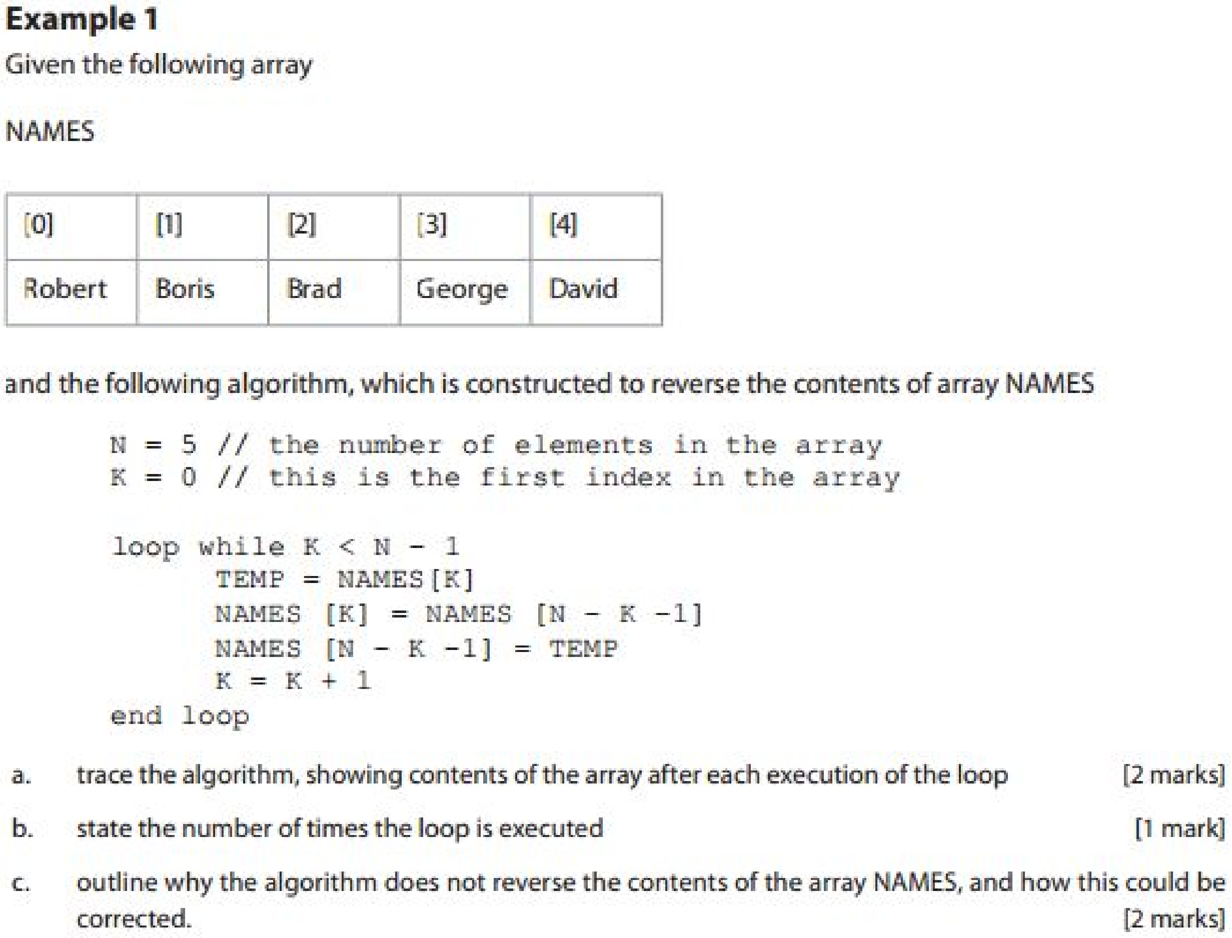
from/to-loop	 COUNT = 0
	SUM = SUM + COUNT
	COUNT = COUNT + 1
	COUNT > 5?
	NO
	YES
		loop COUNT from 0 to 5
		SUM = SUM + COUNT
		end loop

AVERAGING AN ARRAY

The array `STOCK` contains a list of 1000 whole numbers (integers). The following pseudocode presents an algorithm that will count how many of these numbers are non-zero, adds up all those numbers and then prints the average of all the non-zero numbers (divides by `COUNT` rather than dividing by 1000).

```
COUNT = 0
TOTAL = 0

loop N from 0 to 999
  if STOCK[N] > 0 then
    COUNT = COUNT + 1
    TOTAL = TOTAL + STOCK[N]
  end if
end loop

if NOT COUNT = 0 then
  AVERAGE = TOTAL / COUNT
  output "Average = " , AVERAGE
else
  output "There are no non-zero values"
end if
```

[The answers to the the 3 example questions are at the end of this document]



Example 1

Given the following array

NAMES

[0]	[1]	[2]	[3]	[4]
Robert	Boris	Brad	George	David

and the following algorithm, which is constructed to reverse the contents of array NAMES

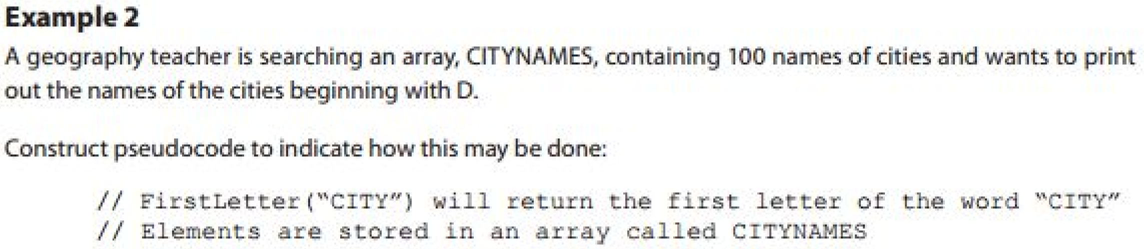
```
N = 5 // the number of elements in the array
K = 0 // this is the first index in the array

loop while K < N - 1
      TEMP = NAMES[K]
      NAMES [K] = NAMES [N - K -1]
      NAMES [N - K -1] = TEMP
      K = K + 1
end loop
```

a. trace the algorithm, showing contents of the array after each execution of the loop [2 marks]

b. state the number of times the loop is executed [1 mark]

c. outline why the algorithm does not reverse the contents of the array NAMES, and how this could be corrected. [2 marks]

[In this example the method to obtain the first letter of a string can be treated abstractly (i.e. the detail of how it works is not relevant, you can just use it)]



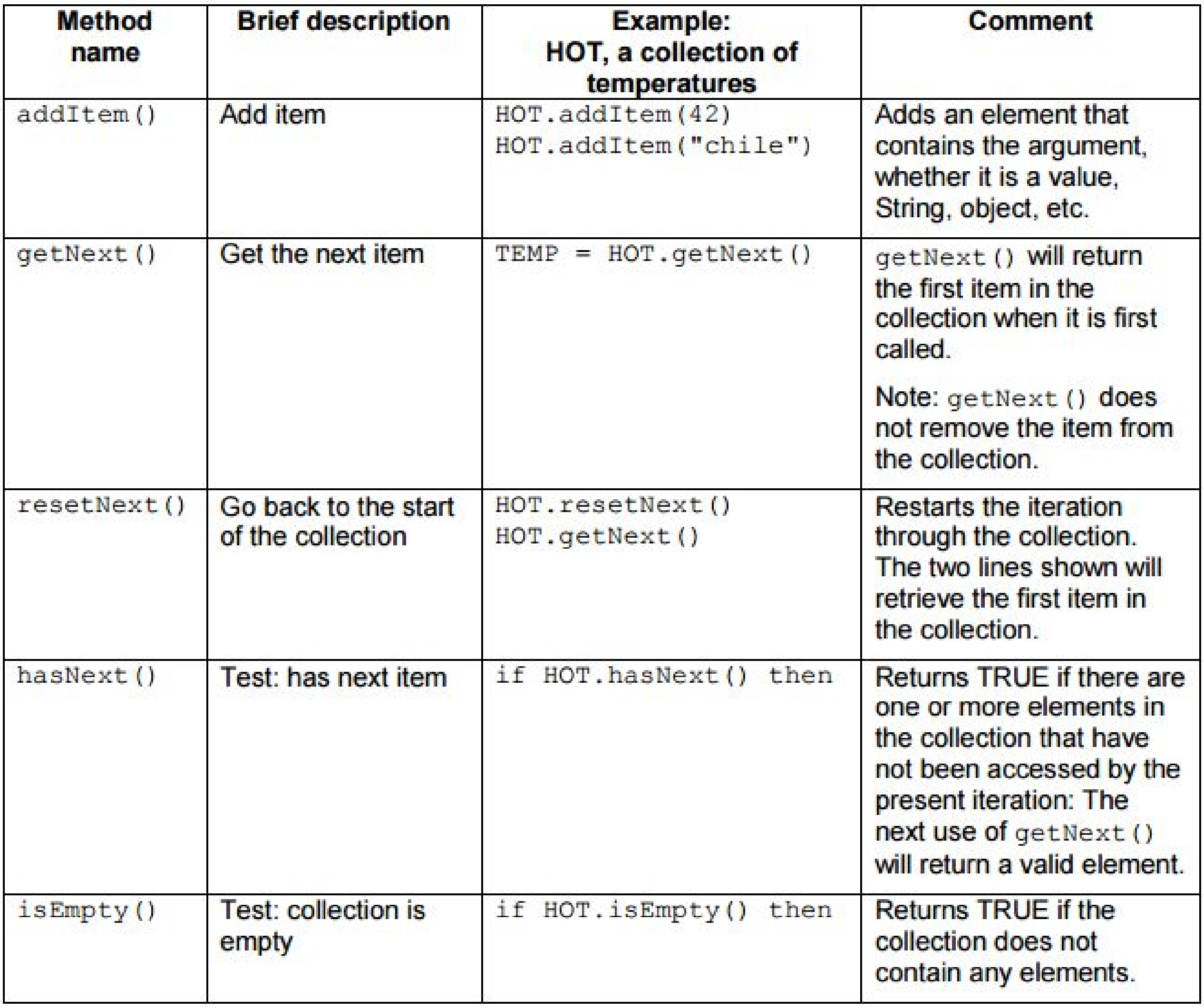
Example 2

A geography teacher is searching an array, CITYNAMES, containing 100 names of cities and wants to print out the names of the cities beginning with D.

Construct pseudocode to indicate how this may be done:

```
// FirstLetter("CITY") will return the first letter of the word "CITY"
// Elements are stored in an array called CITYNAMES
```

COLLECTIONS



Method name	Brief description	Example: HOT, a collection of temperatures	Comment

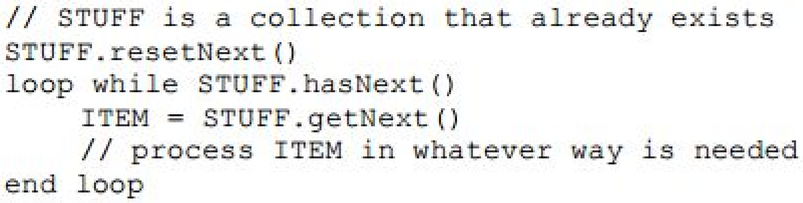
`addItem()`	Add item	`HOT.addItem(42)`
		`HOT.addItem("chile")`
			Adds an element that contains the argument, whether it is a value, String, object, etc.

`getNext()`	Get the next item	`TEMP = HOT.getNext()`	`getNext()` will return the first item in the collection when it is first called. Note: `getNext()` does not remove the item from the collection.

`resetNext()`	Go back to the start of the collection	`HOT.resetNext()`
		`HOT.getNext()`
			Restarts the iteration through the collection. The two lines shown will retrieve the first item in the collection.

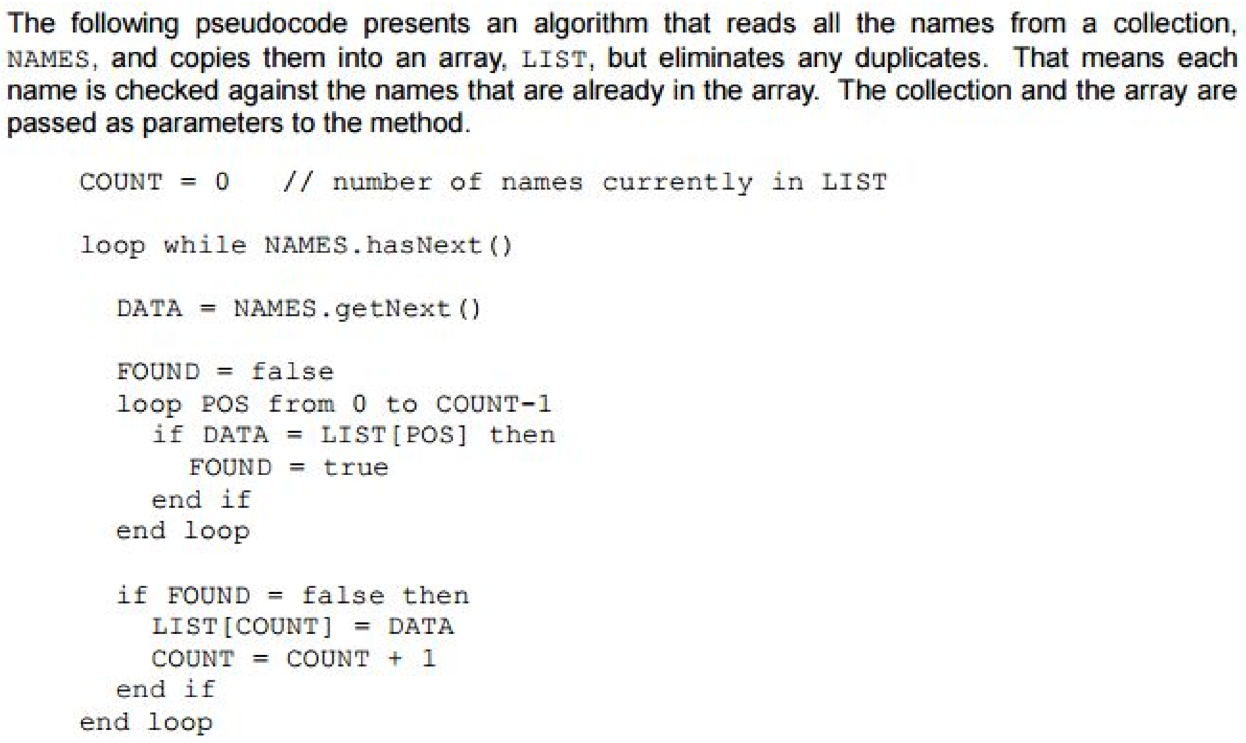
`hasNext()`	Test: has next item	`if HOT.hasNext() then`	Returns TRUE if there are one or more elements in the collection that have not been accessed by the present iteration: The next use of `getNext()` will return a valid element.
`isEmpty()`	Test: collection is empty	`if HOT.isEmpty() then`	Returns TRUE if the collection does not contain any elements.

Looping through a collection



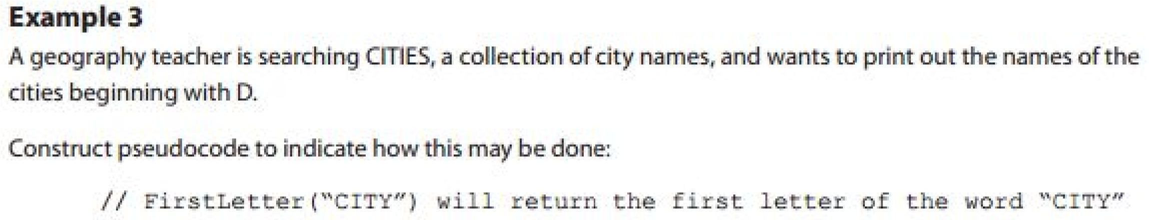
```
// STUFF is a collection that already exists
STUFF.resetNext()
loop while STUFF.hasNext()
    ITEM = STUFF.getNext()
    // process ITEM in whatever way is needed
end loop
```

Looping through a collection to copy to an array [removing duplicates]



The following pseudocode presents an algorithm that reads all the names from a collection, NAMES, and copies them into an array, LIST, but eliminates any duplicates. That means each name is checked against the names that are already in the array. The collection and the array are passed as parameters to the method.

```
COUNT = 0   // number of names currently in LIST

loop while NAMES.hasNext()

  DATA = NAMES.getNext()

  FOUND = false
  loop POS from 0 to COUNT-1
    if DATA = LIST[POS] then
      FOUND = true
    end if
  end loop

  if FOUND = false then
    LIST[COUNT] = DATA
    COUNT = COUNT + 1
  end if
end loop
```



Example 3

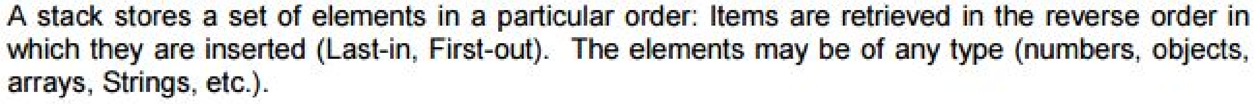
A geography teacher is searching CITIES, a collection of city names, and wants to print out the names of the cities beginning with D.

Construct pseudocode to indicate how this may be done:

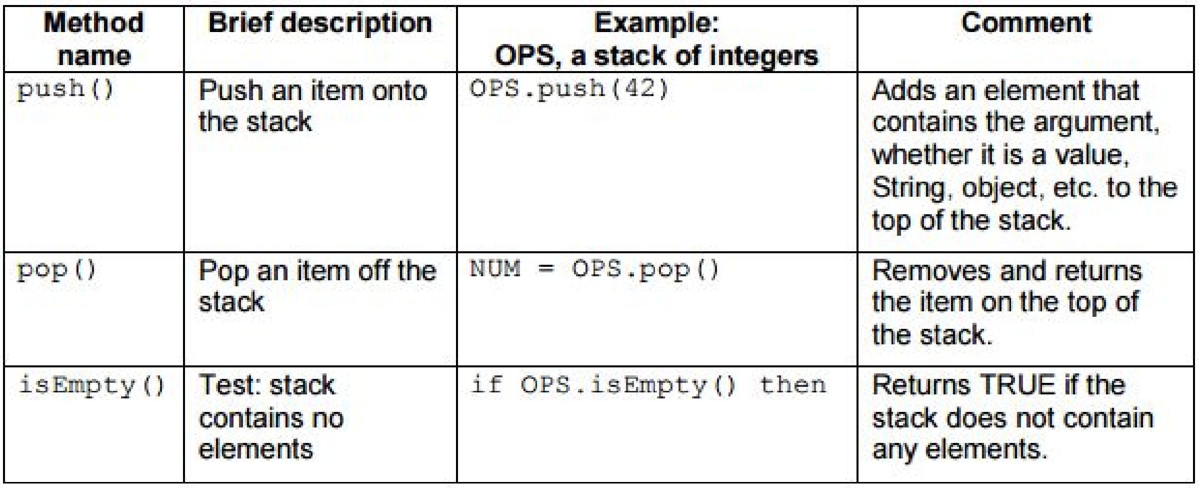
```
// FirstLetter("CITY") will return the first letter of the word "CITY"
```

== **HIGHER LEVEL ONLY SECTION**

STACKS

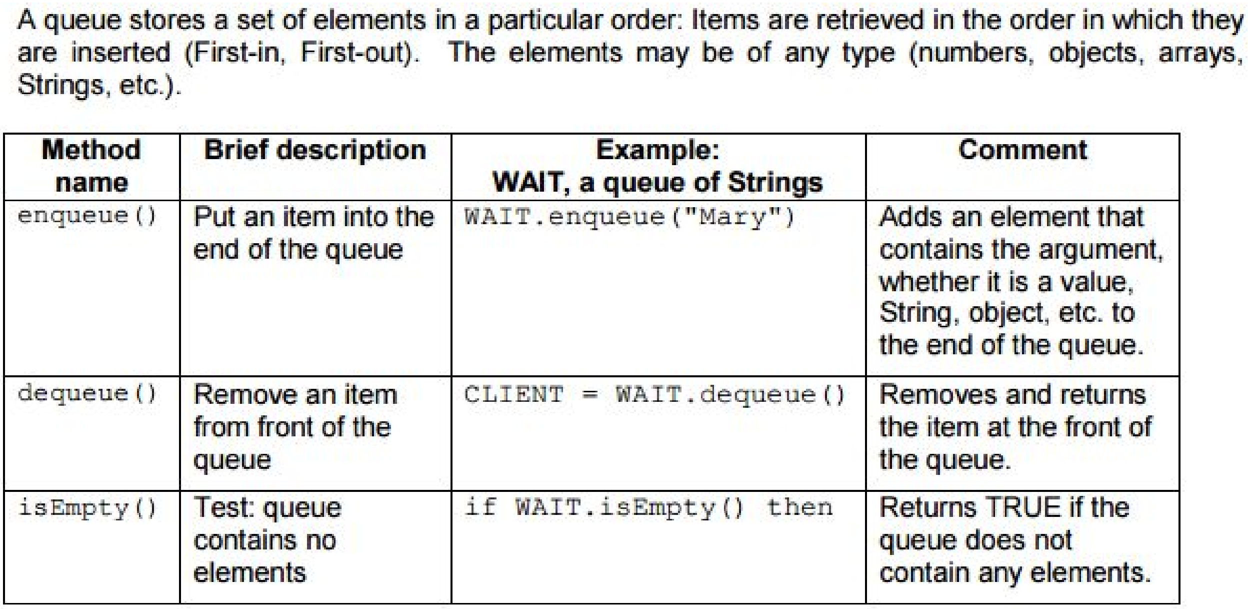


A stack stores a set of elements in a particular order: Items are retrieved in the reverse order in which they are inserted (Last-in, First-out). The elements may be of any type (numbers, objects, arrays, Strings, etc.).



Method name	Brief description	Example: OPS, a stack of integers	Comment
`push()`	Push an item onto the stack	`OPS.push(42)`	Adds an element that contains the argument, whether it is a value, String, object, etc. to the top of the stack.
`pop()`	Pop an item off the stack	`NUM = OPS.pop()`	Removes and returns the item on the top of the stack.
`isEmpty()`	Test: stack contains no elements	`if OPS.isEmpty() then`	Returns TRUE if the stack does not contain any elements.

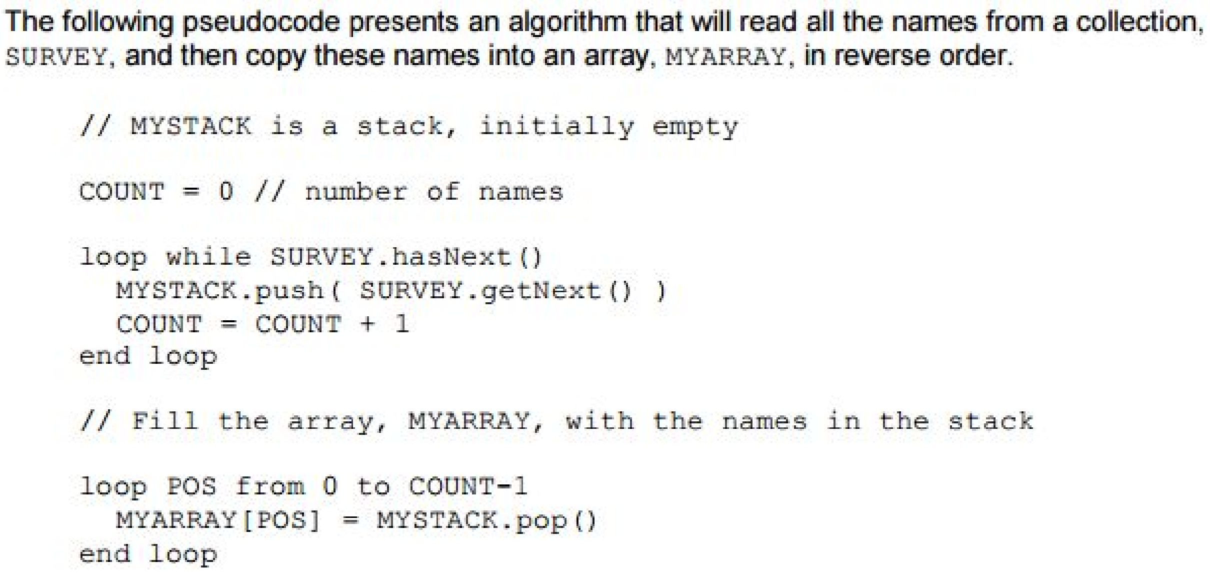
QUEUES



A queue stores a set of elements in a particular order: Items are retrieved in the order in which they are inserted (First-in, First-out). The elements may be of any type (numbers, objects, arrays, Strings, etc.).

Method name	Brief description	Example: WAIT, a queue of Strings	Comment
`enqueue()`	Put an item into the end of the queue	`WAIT.enqueue("Mary")`	Adds an element that contains the argument, whether it is a value, String, object, etc. to the end of the queue.
`dequeue()`	Remove an item from front of the queue	`CLIENT = WAIT.dequeue()`	Removes and returns the item at the front of the queue.
`isEmpty()`	Test: queue contains no elements	`if WAIT.isEmpty() then`	Returns TRUE if the queue does not contain any elements.

USING THE LIFO NATURE OF THE STACK TO REVERSE A COLLECTION INTO AN ARRAY



The following pseudocode presents an algorithm that will read all the names from a collection, SURVEY, and then copy these names into an array, MYARRAY, in reverse order.

```
// MYSTACK is a stack, initially empty

COUNT = 0 // number of names

loop while SURVEY.hasNext()
  MYSTACK.push( SURVEY.getNext() )
  COUNT = COUNT + 1
end loop

// Fill the array, MYARRAY, with the names in the stack

loop POS from 0 to COUNT-1
  MYARRAY[POS] = MYSTACK.pop()
end loop
```

==================================== END OF   HIGHER LEVEL ONLY SECTION

Answers to example questions

Example 1

a) Note the additional columns are not required for the answer but help with the trace

NAMES

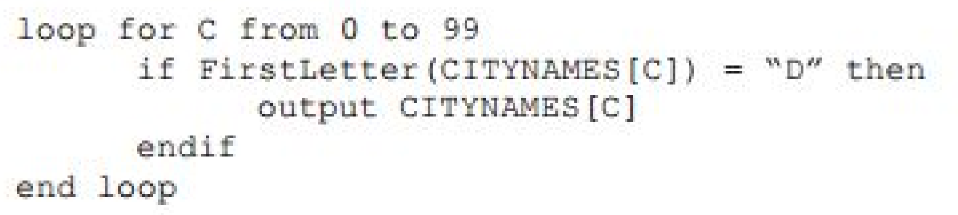
[0]	[1]	[2]	[3]	[4]	K	N	TEMP
Robert	Boris	Brad	George	David	0	5	Robert
David	Boris	Brad	George	Robert	1		Boris
David	George	Brad	Boris	Robert	2		Brad
David	George	Brad	Boris	Robert	3		Boris
David	Boris	Brad	George	Robert	4		

[Loop terminates because K = N - 1 so is no longer < N -1]

b) The loop executes 4 times

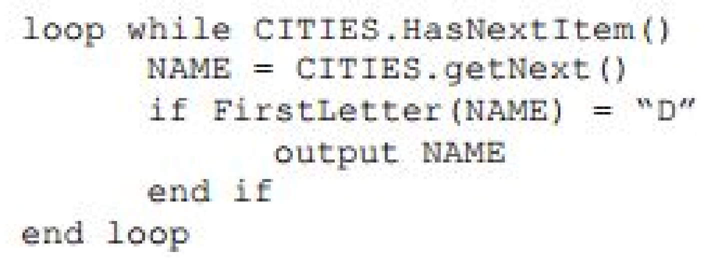
c) The algorithm reverses the array but then keeps looping and re-reverses all but the outer elements of the array, To fix this change the while condition to: K < N div 2

Example 2



```
loop for C from 0 to 99
      if FirstLetter(CITYNAMES[C]) = "D" then
            output CITYNAMES[C]
      endif
end loop
```

Example 3



```
loop while CITIES.HasNextItem()
      NAME = CITIES.getNext()
      if FirstLetter(NAME) = "D"
            output NAME
      end if
end loop
```