Trace Tables(key)

1. Note:A string is an array of characters. For example, in this method the element referred to by STR[9] is the 10th element in the string and STR[0] refers to the first element in the string.

A single space character is represented as " " in the algorithm.

double CALC(String STR)

C = 0

S = 0

T = 0

loop until STR[S] = "."

if STR[S] = " " then

C = C + 1

else

T = T + 1

end if

S = S + 1

end loop

return T / (C + 1)

1. Complete the trace table below for the following call to method CALC(“it is.”).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STR** | **C** | **S** | **T** | **STR[S]** |
| it is. | 0 | 0 | 0 |  |
| it is. | 0 | 1 | 1 | i |
| it is. | 0 | 2 | 2 | t |
| it is. | 1 | 3 | 2 | “ “ |
| it is. | 1 | 4 | 3 | i |
| it is. | 1 | 5 | 4 | s |
| it is. | 1 | 5 | 4 | . |

\*first row are the values of the variables values before entering the loop

1. State the value returned by the method.

4 / (2) = 2.0

1. The following is an algorithm for a method named prime.

boolean PRIME(int NUMBER)

FACTOR = 2

P = false

FOUND = false

loop while (FACTOR \* FACTOR) <= NUMBER AND NOT FOUND

if NUMBER mod FACTOR = 0 then

FOUND = true

end if

FACTOR = FACTOR + 1

end loop

P = NOT FOUND

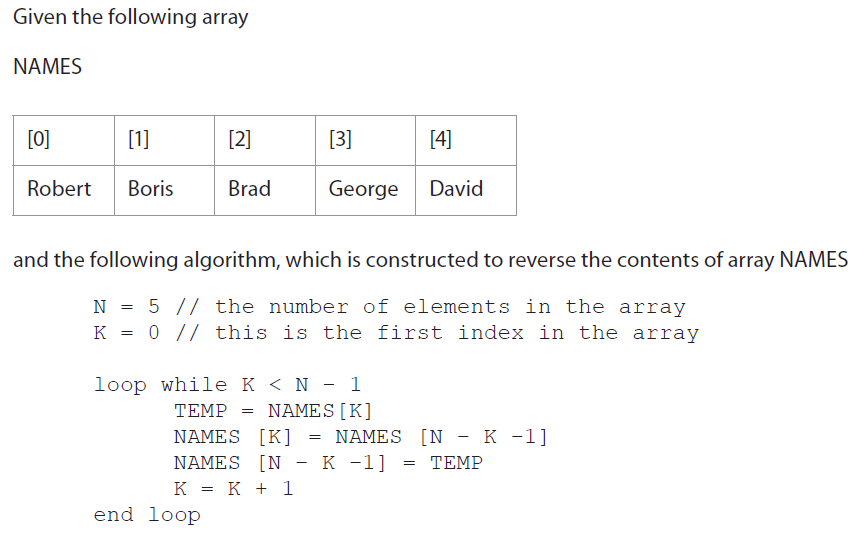
return P

1. Complete the following trace table for the call PRIME(9).

|  |  |  |
| --- | --- | --- |
| **FACTOR** | **FOUND** | **P** |
| 2 | false | false |
| 3 | true | false |
| 4 | true | false |
|  |  |  |
|  |  |  |
|  |  |  |

1. Use trace table to determine the return value when PRIME(20) is called.

|  |  |  |
| --- | --- | --- |
| **FACTOR** | **FOUND** | **P** |
| 2 | false | true |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |



1. Trace the algorithm, showing contents of the array after each execution of the loop.

NAMES

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K | N-K-1 | [0] | [1] | [2] | [3] | [4] |
|  |  | Robert | Boris | Brad | George | David |
| 0 | 4 | **David** | Boris | Brad | George | **Robert** |
| 1 | 3 | David | **George** | Brad | **Boris** | Robert |
| 2 | 2 | David | George | **Brad** | Boris | Robert |
| 3 | 1 | David | **Boris** | Brad | **George** | Robert |

1. State the number of times the loop is executed.

4 times

1. Outline why the algorithm does not reverse the contents of the array NAMES, and how this could be corrected.

Make loop execute 3 times by using while K < N / 2

1. The names of the members of a cycling club are stored in an array ***NAMES*** as shown below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] |
| JONES | SMITH | GOMEZ | SINGH | BUTLER | HU |

After a competition, an array of positions ***POS*** is formed as follows.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] |
| 2 | 4 | 2 | 3 | 1 | 5 |

There was a tie for second place.

Consider the following algorithm fragment. The arrays *NAMES*, *POS*, and *TEMP* are all declared with a size of 6.

loop I from 0 to 5

TEMP[I] = “ZZZ”

end loop

loop I from 0 to 5

TEMP[POS[I]-1] = NAMES[I]

end loop

loop I from 0 to 5

NAMES[I] = TEMP[I]

end loop

1. Complete the following trace table for values 0 to 5 in the second loop of the algorithm.

|  |  |  |
| --- | --- | --- |
| I | POS[I] | TEMP[POS[I]-1] |
| 0 | 2 | Butler |
| 1 | 4 | ~~Jones~~ Gomez |
| 2 | 2 | Singh |
| S | 3 | Smith |
| 4 | 1 | Hu |
| 5 | 5 |  |

1. List the contents of the array NAMES after the third loop has been executed.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **[0]** | **[1]** | **[2]** | **[3]** | **[4]** | **[5]** |
| Butler | Gomez | Singh | Smith | Hu |  |

1. State the purpose of the algorithm.

Arranges names in order by place.

1. Suggest how the problem with the two competitors who tied could be avoided.

loop I from 0 to 5

if TEMP[POS[I]-1] != “ZZZ” then  
 J = TEMP.length – 1  
 loop while J > POS[I]-1  
 TEMP[J] = TEMP[J-1]  
 J--  
 end loop  
 TEMP[POS[I]-1] = NAMES[I];

else

TEMP[POS[I]-1] = NAMES[I];

end if

end loop

1. Look at the following algorithm.

TOTAL = 0

loop I from 0 to 4

loop J from 0 to I

TOTAL = TOTAL + 1

end loop

end loop

output TOTAL

1. Construct a trace table for the variables I, J, TOTAL.

|  |  |  |
| --- | --- | --- |
| I | J | Total |
| 0 | 0 | 1 |
| 1 | 0 | 2 |
| 1 | 1 | 3 |
| 2 | 0 | 4 |
| 2 | 1 | 5 |
| 2 | 2 | 6 |
| 3 | 0 | 7 |
| 3 | 1 | 8 |
| 3 | 2 | 9 |
| 3 | 3 | 10 |
| 4 | 0 | 11 |
| 4 | 1 | 12 |
| 4 | 2 | 13 |
| 4 | 3 | 14 |
| 4 | 4 | 15 |

1. What is the value of TOTAL when output?

15 – it’s a summation (1 + 2 + 3 + 4 + 5 = 15)