**M104**

**Assessment Workshop 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student No** |  | **Name** |  |

* Study the following algorithm:

In an array, each pair of adjacent elements is compared, and the elements are swapped if they are not in order.  For ascending order, the upper element must be smaller than the lower element.

Complete the following table for the comparison work:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | Comparison | | | |  |
| No | Array Content | 1 | 2 | 3 | 4 | Final |
| 1 | 83 | **5** | 5 | 5 | 5 | 5 |
| 2 | 5 | **83** | 21 | 21 | 21 | 21 |
| 3 | 21 | 21 | 83 | 14 | 14 | 14 |
| 4 | 14 | 14 | 14 | 83 | 77 | 77 |
| 5 | 77 | 77 | 77 | 77 | 83 | 83 |

* Use the following program skeleton, complete the program named *BubbleSort.java* to achieve the above requirements. The array elements can be hardcoded with the given numbers.

public class BubbleSort {

public static void main(String [ ] args) {

int [ ] arr = {83, 5, 21, 14, 77};

int temp;

*// handling sorting*

for (int i=0; i<????????; i++)

{

// handling the swapping

}

*// printing the result*

for (int i=0; i<??????; i++)

System.out.print(arr[i] + " ");

System.out.println();

}

}

Do you know if the array contains *N* elements how many comparisons are required?

\_\_\_\_\_\_N \*N\_\_\_\_\_\_\_\_\_\_\_\_\_

* Repeat the same comparison process, mentioned in 1, for many rounds. Fill in the result:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Round 2 | | | | | | |
|  |  | Comparison | | | |  |
| No | Array Content | 1 | 2 | 3 | 4 | Final |
| 1 | 5 | 5 |  |  |  | 5 |
| 2 | 21 | 21 | 14 |  |  | 14 |
| 3 | 14 | 14 | 21 | 21 |  | 21 |
| 4 | 77 | 77 |  | 77 | 83 | 77 |
| 5 | 83 | 83 |  |  | 83 | 83 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Round 3 | | | | | | |
|  |  | Comparison | | | |  |
| No | Array Content | 1 | 2 | 3 | 4 | Final |
| 1 | 5 | 5 |  |  |  | 5 |
| 2 | 14 | 14 | 21 |  |  | 14 |
| 3 | 21 |  | 21 | 77 |  | 21 |
| 4 | 77 |  |  | 77 | 88 | 77 |
| 5 | 83 |  |  |  | 88 | 83 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Round 4 | | | | | | |
|  |  | Comparison | | | |  |
| No | Array Content | 1 | 2 | 3 | 4 | Final |
| 1 |  | 5 |  |  |  | 5 |
| 2 |  | 14 | 21 |  |  | 14 |
| 3 |  |  | 21 | 77 |  | 21 |
| 4 |  |  |  | 77 | 83 | 77 |
| 5 |  |  |  |  | 83 | 83 |

* Do you know if the array contains *N* elements how many rounds are required?

N-1

* Complete the program so that a proper sorting sequence could be obtained.
* This sorting algorithm is called *Bubble Sort*. Could you observe why it is so called?

cuase it compare two elements and the heavy one get lower in array and the and the light one raise to lower

* Check whether you could shorten the comparison process?

No i cant

* In case the elements are already in the sequence, would your program still need to go through the same number of passes (rounds)? Include in your program to make the work more efficient for this scenario?

My code check if the elements after every round and if it in sequence it stops

*Paste your program source below*

package com.company;  
  
import java.util.Scanner;  
  
public class Main {  
  
  
 static void printArray (int[] array ) {  
 for (int i = 0; i < array.length; ++i)  
 System.*out*.println(" "+ array[i]);  
 }  
  
 public static void main(String[] args) {  
  
  
 boolean isSorted = true;  
 int[] a ;  
 a = new int[]{83, 5, 21, 14, 77};  
 int n, i, j, temp, k;  
 Scanner scanner = new Scanner(System.*in*);  
  
  
 System.*out*.println("the unsorted array ");  
 *printArray*(a);  
 i = 1;  
 while ((i<a.length-1) && (isSorted = true))  
 {  
 isSorted = false;  
 for (j = 0; j<(a.length - i); ++j){  
 if (a[j]>a[j + 1])  
 {  
 temp = a[j];  
 a[j] = a[j + 1];  
 a[j + 1] = temp;  
 isSorted = true;  
 }  
 }  
 if (isSorted== false)break;  
 System.*out*.println("Round "+ " "+ i);  
 *printArray*(a);  
  
 i++;  
  
 }  
 System.*out*.println( "Array after bubble sort:");  
 *printArray*(a);  
 }  
   
}

*Screen capture your output result below*

|  |
| --- |
|  |

* Even after your tuning in the algorithm, do you think that *bubble sort* is a good efficient sorting mechanism? Comments.

|  |
| --- |
| No it is not a good cuase it take N\*N time complicity |

* Submit:
* Your final version of Bubble Sort java program
* This document with answers

to the SOUL platform.