**Northeastern Illinois University**

**CS-200: Programming I**

**PLTL: Nested Loops**

**Problem 1**

* Write a program that has the class name Problem1 and that has the main method. Leave

the main method empties for now.

* Write a method named sort that takes one parameter, an integer array a and does not

return anything.

* The method should modify the array a by putting its elements in increasing order (smallest to largest).
* Create a printArray method that takes a character array as a parameter and prints out

the elements of the array on the same line separated by a space.

* Several sample usages are provided for you below. Use the sample usages in the main

method to test your code (and use the printArray method to print out the results of

calling the sort method!).

|  |  |
| --- | --- |
| Sample method usage | Value of input array after method call |
| **int** [] b = { -1, 10, 4, 8};  sort(b); | { -1, 4, 8, 10} |
| **int** [] c = {5, -3};  sort(c); | { -3, 5} |
| **int** [] d = {9, 9, 2, 5, 0};  sort(d); | {0, 2, 5, 9, 9} |

**Problem 2**

* Write a program that has the class name Problem2 and that has the main method. Leave

the main method empties for now.

* Write a method named noDuplicates that takes one parameter, an integer array a and

returns a new integer array.

* The method should create and return a new 1D array of integers with any duplicate values from a removed. Do not modify the input parameter a.
* Create a printArray method that takes a character array as a parameter and prints out

the elements of the array on the same line separated by a space.

* Several sample usages are provided for you below. Use the sample usages in the main

method to test your code (and use the printArray method to print out the results of

calling the noDuplicates method!).

|  |  |
| --- | --- |
| Sample method usage | Value of input array after method call |
| **int** [] b = {2, 2};  i**nt** [] b1 = noDuplicates(b); | {2} |
| **int** [] c = {1, 5, -4, 5, -4, 1, 0};  **int** [] c1 = noDuplicates(c); | {1, 5, -4, 0} |
| **int** [] d = { -2, 8, 9};  **int** [] d1 = noDuplicates(d); | { -2, 8, 9} |

**Problem 3**

* Write a program that has the class name Problem3 and that has the main method.
* Write a program that asks a user to enter the length of a square greater than 1.
* Prompt the user to enter the length until they enter number greater than 1.
* The program should create a full Square using length.
* Your output must match the sample output format exactly.

|  |
| --- |
| Enter a square length greater than 1 (> 1): 5  5 \* \* \* 1  \* 4 \* 2 \*  \* \* 3 \* \*  \* 4 \* 2 \*  5 \* \* \* 1 |

|  |
| --- |
| Enter a square length greater than 1 (> 1): -1  Enter a square length greater than 1 (> 1): 1  Enter a square length greater than 1 (> 1): 3  3 \* 1  \* 2 \*  3 \* 1 |

**Problem 4**

* Write a program that has the class name Problem4 and that has the main method. Leave

the main method empties for now.

* Write a method named closestPower that takes two parameter, an integer n and an integer

array a and returns an integer.

* The method should find num which is the value in array a that appears the most. If none

of the value is repeated then consider the largest one as num.

* Now that you have num and n, find the integer i which is the power of n, such that n is

closest to num and return the value of i.

* In case of tie, return the smaller value.
* Several sample usages are provided for you below. Use the sample usages in the main

method to test your code.

|  |  |
| --- | --- |
| Sample Method Usage | Return Value |
| **int** n1 = 3;  **int** [] a1 = {6, 81, 17, 12, 25, 24, 12};  **int** x1 = closestPower (n1, a1); | 2 |
| **int** n2 = 4;  **int** [] a2 = {3, 4, 5, 1, 12, 67, 3, 1, 1};  **int** x2 = closestPower (n2, a2); | 0 |
| **int** n3 = 7;  **int** [] a3 = {77, 22, 185, 20, 269, 88};  **int** x3 = closestPower (n3, a3); | 3 |
| **int** n4 = 2;  **int** [] a4 = {1, 4, 24, 3, 12, 8};  **int** x2 = closestPower (n4, a4); | 4 |

**Problem 5**

* Write a program that has the class name Problem5 and that has the main method. Prompt

the user to enter an integer that represents row and column values. You can assume that

the user will enter a positive value.

* The program should create a multiplication table by finding the product of the row and

column, beginning at 1 and going up to the row or column value. The table should end

at row \* column.

* Several sample usages are provided for you below.

|  |
| --- |
| Enter row: 4  Enter col: 7  1 2 3 4 5 6 7  2 4 6 8 10 12 14  3 6 9 12 15 18 21  4 8 12 16 20 24 28 |

|  |
| --- |
| Enter row: 1  Enter column: 5  1 2 3 4 5 |

|  |
| --- |
| Enter row: 3  Enter column: 4  1 2 3 4  2 4 6 8  3 6 9 12 |

**Problem 6**

* Write a program that has the class name Problem6 and that has the main method.
* Write a program that creates a 10 by 10 box. The box should look exactly as the sample

usage below.

* Even though the values are given, nothing should be hard coded - all constants must be

assigned a variable.

* Several sample usages are provided for you below.

|  |
| --- |
| 1 2 3 4 5 6 7 8 9 10  2 20  3 30  4 40  5 50  6 60  7 70  8 80  9 90  10 20 30 40 50 60 70 80 90 100 |