Algorithm Programs

IMPORTANT NOTE - Create a Util Class and write the Anagram, Palindrome, Prime Numbers, Search Algos, Sort Algos, etc as a static function

1. **An Anagram Detection Example**
    a. Desc -> One string is an anagram of another if the second is simply a rearrangement of the first. For example, 'heart' and 'earth' are anagrams...
    b. I/P -> Take 2 Strings as Input such abcd and dcba and Check for Anagrams
    c. O/P -> The Two Strings are Anagram or not....
2. Take a range of 0 - 1000 Numbers and find the Prime numbers in that range.
3. Extend the above program to find the prime numbers that are Anagram and Palindrome
4. **To the Utility Class write the following static methods**
    a. Desc -> Create Utility Class having following static methods
        i. binarySearch method for integer
        ii. binarySearch method for String
        iii. insertionSort method for integer
        iv. insertionSort method for String
        v. bubbleSort method for integer
        vi. bubbleSort method for String
    b. I/P -> Write main function to call the utility function
    c. Logic -> Check using Stopwatch the Elapsed Time for every method call
    d. O/P -> Output the Search and Sorted List. More importantly print elapsed time performance in descending order
5. **Question to find your number**
    a. Desc -> takes a command-line argument N, asks you to think of a number between 0 and N-1, where N = 2^n, and always guesses the answer with n questions.
    b. I/P -> the Number N and then recursively ask true/false if the number is between a high and low value
    c. Logic -> Use Binary Search to find the number
    d. O/P -> Print the intermediary number and the final answer

6. **Binary Search the Word from Word List**

    a. Desc -> Read in a list of words from File. Then prompt the user to enter a word to search the list. The program reports if the search word is found in the list.
    b. I/P -> read in the list words comma separated from a File and then enter the word to be searched
    c. Logic -> Use Arrays to sort the word list and then do the binary search
    d. O/P -> Print the result if the word is found or not

7. **Insertion Sort**

    a. Desc -> Reads in strings from standard input and prints them in sorted order. Uses insertion sort.
    b. I/P -> read in the list words
    c. Logic -> Use Insertion Sort to sort the words in the String array
    d. O/P -> Print the Sorted List

8. **Bubble Sort**

    a. Desc -> Reads in integers prints them in sorted order using Bubble Sort
    b. I/P -> read in the list ints
    c. O/P -> Print the Sorted List

9. **Merge Sort** - Write a program with Static Functions to do Merge Sort of list of Strings.
    a. Logic -> To Merge Sort an array, we divide it into two halves, sort the two halves independently, and then merge the results to sort the full array. To sort a[lo, hi), we use the following recursive strategy:
    b. Base case: If the subarray length is 0 or 1, it is already sorted.
    c. Reduction step: Otherwise, compute mid = lo + (hi - lo) / 2, recursively sort the two subarrays a[lo, mid) and a[mid, hi), and merge them to produce a sorted result.

10. **Find the Fewest Notes to be returned for Vending Machine**

    a. Desc -> There is 1, 2, 5, 10, 50, 100, 500 and 1000 Rs Notes which can be returned by Vending Machine. Write a Program to calculate the minimum number of Notes as well as the Notes to be returned by the Vending Machine as a Change
    b. I/P -> read the Change in Rs to be returned by the Vending Machine
    c. Logic -> Use Recursion and check for largest value of the Note to return change to get to minimum number of Notes.

d. O/P -> Two Outputs - one the number of minimum Note needed to give the change and second list of Rs Notes that would given in the Change

11. To the Util Class add ***dayOfWeek*** static function that takes a date as input and prints the day of the week that date falls on. Your program should take three command-line arguments: m (month), d (day), and y (year). For m use 1 for January, 2 for February, and so forth. For output print 0 for Sunday, 1 for Monday, 2 for Tuesday, and so forth. Use the following formulas, for the Gregorian calendar (where / denotes integer division):

$y_0 = y - (14 - m) / 12$

$x = y_0 + y_0/4 - y_0/100 + y_0/400$

$m_0 = m + 12 \times ((14 - m) / 12) - 2$

$d_0 = (d + x + 31m_0 / 12) \bmod 7$

12. To the Util Class add ***temperaturConversion*** static function, given the temperature in fahrenheit as input outputs the temperature in Celsius or viceversa using the formula

Celsius to Fahrenheit: (°C × 9/5) + 32 = °F

Fahrenheit to Celsius: (°F − 32) x 5/9 = °C

13. Write a Util Static Function to calculate ***monthlyPayment*** that reads in three command-line arguments P, Y, and R and calculates the monthly payments you would have to make over Y years to pay off a P principal loan amount at R per cent interest compounded monthly. The formula is The formula is

```
                  P r
payment =   ---------------,   where n = 12 * Y, r = R / (12 * 100)
            1  - (1 + r)^(-n)
```

14. Write a static function ***sqrt*** to compute the square root of a nonnegative number c given in the input using Newton's method:

- initialize t = c

- replace t with the average of c/t and t

- repeat until desired accuracy reached using condition Math.abs(t - c/t) > epsilon*t where epsilon = 1e-15;

15. Write a static function ***toBinary*** that outputs the binary (base 2) representation of the decimal number typed as the input. It is based on decomposing the number into a sum of powers of 2. For example, the binary representation of 106 is 11010102, which is the same as saying that 106 = 64 + 32 + 8 + 2. Ensure necessary padding to represent 4 Byte String.

To compute the binary representation of n, we consider the powers of 2 less than or equal to n in decreasing order to determine which belong in the binary decomposition (and therefore correspond to a 1 bit in the binary representation).

16. Write Binary.java to read an integer as an Input, convert to Binary using toBinary function and perform the following functions.

i. Swap nibbles and find the new number.

ii. Find the resultant number is the number is a power of 2.

A nibble is a four-bit aggregation, or half an octet. There are two nibbles in a byte.

Given a byte, swap the two nibbles in it. For example 100 is to be represented as 01100100 in a byte (or 8 bits). The two nibbles are (0110) and (0100). If we swap the two nibbles, we get 01000110 which is 70 in decimal.

17. You have a long list of tasks that you need to do today. To accomplish task you need M minutes, and the deadline for this task is D . You need not complete a task at a stretch. You can complete a part of it, switch to another task, and then switch back.You've realized that it might not be possible to complete all the tasks by their deadline. So you decide to do them in such a manner that the maximum amount by which a task's completion time overshoots its deadline is minimized.

Input Format

The first line contains the number of tasks, . Each of the next lines contains two integers,D and M .

Output Format

Output *T* lines. The *ith* line contains the value of the maximum amount by which a task's completion time overshoots its deadline, when the first tasks on your list are scheduled optimally.