Details

Java programs with good logic, comments and algorithms related to 11th and 12th ISC CS Syllabus.

License

Format

1. Front Page
2. Acknowledgement
3. Algorithms
4. Code with Comments
5. Outputs

Topics

1. If Else
2. Loops
3. Menu Driven
4. Functions
5. Constructors
6. Arrays - 1D, 2D
7. Strings
8. String Tokenizer
9. Object Passing
10. Files

11. Errors and Exceptions
12. Recursion
13. Inheritance
14. LinkedList
15. Stack
16. Queue

Index

1. achillesNumber
2. adder
3. arrayTools
4. arrShiftSortMerge
5. binaryDecimalShenanigans
6. blinsMaker
7. boundarySort
8. camelConc
9. chess
10. circularQueue
11. concentricCircles
12. dateCalculator
13. disariumNumber
14. eulerFunction
15. fascinatingNumber
16. goldbachNumber
17. isogramSentence
18. keyboardAnalyser
19. luckyNumbers
20. potential
21. recursiveSortSearch
22. reverseEncryption
23. reversingLinkedList
24. sentencePalindromeGenerator
25. ticTacToe
26. towerOfHanoi
27. typingTest
28. verticalBanners
29. wordleStartChecker
30. wordShift

Algorithm

1. achillesNumber

Topics: Functions, Loops, LinkedList
Algorithm:

1. Start Algorithm
2. Input number to check

3. Check if number if powerful and perfect in two boolean functions returning true or false values in the main method and print if the number is a achilles number or not
4. In function isPerfect, run a for loop from 2 till less than the number and have a nested for loop inside running from j = 2 till less than num. Check if i raised to j is equal to num. If true, return true, else return false. Using this method, check every possibility of two numbers less than the original number that when the second is raised to the first, it gives the original number.
5. In function isPowerful, declare an integer Linked list lk. Find all unique prime factors of n using the following method:
   a. Check if the number is divisible by 2. If true, add 2 to the linked list. Now keep dividing 2 from n while the number is even. After this, the number will be odd.
   b. Run a for loop from i = 3 till square root of n and increment by 2. Check if n is divisible by i. If true, add i to the linked list and keep dividing it from n till i is no longer a factor of n.
6. Now the linked list contains every unique prime factor of n. Run a for loop from i = 0 till lk.size() and check if the square of every element is also a factor of the original number. If true, return true, else return false.
7. End algorithm

2. adder

Topics: Call by Reference, Functions, Constructors
Algorithm:

1. Start algorithm
2. Make two 1d arrays and store hours in index [0] and minutes in index [1]
3. In function readtime, declare scanner object and take input for the calling object the values of minutes and hours and store in a[].
4. In function addtime, pass two objects X and Y. Using the calling object, add the values of hours and minutes. If sum of minutes is > 60 then add 1 to hours and take remainder of minutes in sum
5. In function disptime, print final hours and minutes of calling function.
6. End Algorithm

3. arrayTools

Topics: Functions, 2D Arrays
Algorithm:

1. Start algorithm
2. int[][] input()
    1. Take inputs for number of rows in r, columns in c.
    2. Declare and initialise int[][] arr with given size.
    3. Start for loop from i = 0 till less than r with increment of 1. Inside, start for loop from j = 0 till less than c with increment of 1. Inside, take inputs for all elements of matrix.
3. print()
    1. Start for loop from i = 0 till less than r with increment of 1. Inside, print each 1D row array of arr[i].
4. int[][] transpose()
    1. Store number of rows in r and columns in c.
    2. Declare int[][] trans with size [c][r]

    3. Start for loop from i = 0 till less than c with increment of 1. Inside, start for loop from j = 0 till less than r with increment of 1. Inside, assign trans[i][j] to arr[j][i].
    4. Print the transposed array
5. int[][] transpose()
    1. Declare int[][] trans with size [c][c]
    2. Start for loop from i = 0 till less than c with increment of 1. Inside, start for loop from j = 0 till less than r with increment of 1. Inside, assign trans[i][j] to arr[j][i].
    3. Print the transposed array
6. void rotate(int[][] arr)
    1. Declare and initialise int[][] rot with same size as arr.
    2. Run a for loop with i = 2, k = 2 till i less than rot.length with i increment by 1 and k decrement by 1.
    3. Inside, run a for loop with j = 0 till less than rot[i].length and increment by 1.
    4. Inside, assign rot[j][k] to arr[i][j]
    5. Exit both loops and print rot.
7. void multiply1(int[][] a, int b)
    1. Run a for loop with i = 0 till i less than a.length with i increment by 1 .
    2. Inside, run a for loop with j = 0 till less than a[i].length and increment by 1.
    3. Inside, multiple each element of a[i][j] with constant passed - b.
    4. Exit both loops and print a[][].
8. void addSubtract(int[][] a, int[][] b)
    1. Run a for loop with i = 0 till i less than a.length with i increment by 1.
    2. Declare and initialise arrays int[][] sum and int[][] difference of same sizes as a and b.
    3. Inside, run a for loop with j = 0 till less than a[i].length and increment by 1.
    4. Inside, assign values to sum and difference as the sum and differences of elements a[i][j] and b[i][j].
    5. Exit both loops and print the arrays sum and difference.
9. void arraySums(int[][] a, int r, int c)
    1. Run a for loop with i = 0 till i less than a.length with i increment by 1 .
    2. Inside, run a for loop with j = 0 till less than a[i].length and increment by 1.
    3. Inside, check if i == j. If true, the element is present on the left diagonal. Print it and add the value to a sum variable.
    4. Exit both loops and print sum.
    5. Run a for loop with i = 0 till i less than a.length with i increment by 1 .
    6. Inside, run a for loop with j = 0 till less than a[i].length and increment by 1.
    7. Inside, check if i + j == r - 1. If true, the element is present on the right diagonal. Print it and add the value to a sum variable.
    8. Exit both loops and print sum.

4. arrShiftSortMerge

1. Start algorithm
2. Take inputs for size of first array in n1, values of elements of the array in a[], size of second array in n2, values of elements of the array in b[]
3. Declare an array merge of size n1+n2. Declare variables count for merge, aC for a[] and bC for b[]
4. Start for loop from i = 0 till less than merge.length with increment of 1.

    1. Check if the counters aC and bC have exceeded length of their respective arrays. If true, find the one between aC and bC is greater. When found, dump the remaining contents of the other array in merge[].
    2. Check if (a[aC] < b[bC]). If true, set merge[count++] to a[aC] and increment aC.
    3. Check if (b[bC] < a[aC]). If true, set merge[count++] to b[bC] and increment bC.
    4. Check if (a[aC] == b[bC]). If true, set merge[count] = a[aC], merge[++count] = b[bC] and increment aC and bC by 1.
    5. End for loop
5. Start new for loop from i = 0 till less than merge.length with increment of 1. Print all the values of the array merge[].

5. binaryDecimalShenanigans

Topics: String, Loops, Recursion, If Else
Algorithm:

1. Store the string value of passed integer in StringBuffer. Reverse it and store this value in a string called binary.
2. Declare int decimal and initialise with default value
3. Run a for loop iterating through string binary.
    1. if the character at index i is a '1', increase the value of int decimal with 2^i
4. Return decimal
5. binaryToDecimalRecursion(int n)
    1. Set base case as n == 0. If true, return 0.
    2. Call the recursive statement return n % 10 + 2 * binaryToDecimalRecursion(n / 10). This takes the last digit and adds the twice of the number without the last digit, and continues to do so till n is 0. In the end of this expanded expression of tailwind recursive recursion, 0 is added.
6. decimalToBinary(int decimal)
    1. Declare string binary
    2. Run while loop with condition(decimal > 1)
    3. Inside, store the remainder upon dividing decimal by 2 in string binary and divide decimal by 2.
    4. Return the Integer.valueOf() of a stringbuffer variable with value of string binary, reversed.
7. decimalToBinaryRecursion(int n)
    1. Set base case as n == 0. If true, return 0.
    2. Call the recursive statement return n % 2 + 10 * binaryToDecimalRecursion(n / 10). This takes the remainder of number upon dividing by 2, and, adds ten times of the number without the last digit, and continues to do so till n is 0. In the end of this expanded expression of tailwind recursive recursion, 0 is added.

6. blinsMaker

Topics: If Else
Algorithm:

1. Start algorithm

2. Initialise scanner object and declare variables to store eggs, milk, flour and minimum required to make one portion of blins (pancakes).
3. Calculate the amount of pancakes (blins) one can make using raw ingredients by first taking the available amounts of ingredients present with user.
4. Find number of portions possible by dividing given amount by least amount needed for 1 portion. If the user has less ingredients than what is required to make a single portion, print that they can not make a single portion and exit.
5. Calculate excess ingredients which will not be used for portions by subtracting given amount by (minimum * amount of least ingredient)
6. Print the number of blins / portions the user can make along with the amount of ingredients used and left.
7. End Algorithm.

7. boundarySort

Topics: 2D Array, Loops, Sorting
Algorithm:

1. Start algorithm
2. Take input of number of rows and columns and then take input of the elements in the matrix by running a for loop from i = 0 till rows and nested for loop from j = 0 till columns.
3. Run a for loop from i = 0 till rows and nested for loop from j = 0 till columns and store boundary elements (row or column number is either 0 or length - 1) in a 1d array
4. Sort array using any sorting technique (swap selection sort used here). In selection sort, find the smallest of the row and assign it to a temp variable and swap the initial index with it. Next time, start from the next index and find and swap the next smallest value in the array. Do this till all passes are complete.
5. Run a for loop from i = 0 till rows and nested for loop from j = 0 till columns and put elements back in original matrix manually going clockwise.
6. End Algorithm

8. camelConc

Topics: If Else, Loops, Strings
Algorithm:

1. Start algorithm
2. Take input for String sentence from user
3. Run a for loop from i = 0 till length and seperate the words from the sentence and store them in str[] - in the loop, if sentence[i] is a space, the previous characters form the word. Reset the word variable and move on to the next iteration. Store all words in str[].
4. Run a for loop from i = 0 till str.length and concatenate the words in a final string. Add the first index to the final string all in lowercase and add following words from array to final string by making first character of each word uppercase.
5. Print finalStr.
6. End Algorithm

9. chess

Topics: If Else, Menu
Algorithm:

1. Start algorithm
2. Take input for start and end indexes in the form of x and y coordinates for the cartesian plane like chess board.
3. After analysis of moves that can be made by various chess pieces, they can be described using the geometrical distance formula: pawn can move forward by 1, king can move by 1 in any direction, bishop can move in multiples of √2, rook can move in multiples of integers (1, 2, ...), knight can move in multiples of √5 and queen can move in a combination of the rook and bishop.
4. Find distance between two points on the matrix by doing ((y2 - y1)^2 + (x2 - x1)^2)^0.5
5. Check distances to find which pieces can move.
6. Print values.
7. End Algorithm

10. circularQueue

Topics: Queue, Functions
Algorithm:

1. Start Algorithm
2. main()
    1. Take inputs for max size, true size and all elements.
    2. Run pop() to pop an element from the front
    3. Take input for an element and run push() to push an element to rear
3. isFull()
    1. Return the boolean value of comparison of c and arr.length
4. isEmpty()
    1. Return the boolean value of comparison of c and 0
5. pop()
    1. Check if circular queue is empty. If true, print MIN_VALUE
    2. Store the value of f % c in f.
    3. Return arr[++f]
6. push(int n)
    1. Check if circular queue is full. If true, print MAX_VALUE
    2. Store the value (r+1) % arr.length in r
    3. Store value of arr[r] in int n
    4. Increment the value of c by 1
    5. if(f == -1), store value of r in int f
7. circularQueue(int size)
    1. Set default values of int f and r to -1 and c to 0.
    2. Store value of passed size in instance variable size.

11. concentricNumbers

Topics: Loops
Algorithm:

1. Start algorithm
2. The 2d array will always be a square of radius 2n - 1 where n is input

3. Run two for loops covering every number in the 2d array. Find the larger difference -> row number - mid index or column number - mid index
4. Print the bigger number + 1 as the integer for that place in the matrix to complete the pattern
5. The distance formula is disected here into the vertical and horizontal component. The bigger value is used instead of adding the squares and taking the square root.
6. End Algorithm

12. dateCalculator

Topics: Arrays, Functions, Loops
Algorithm:

1. Start Algorithm
2. Define an Array containing days number leaving blank(0th position)
3. Define another Array containing Month names leaving blank(0th position)
4. Now Input Day Number, N number of days, Year from the user
5. In a variable store the sum of Day number and N number (for calculating date after n days)
6. Now check if the year entered is leap year or not
7. If it is leap year than increment the days number in the array from 2nd position by 1
8. Now check if the days entered are in the range or not
9. If the days are in the range than calculate the date (date can be calculated by subtracting the entered days number from the original days number)
10. If the days are not in the range than change the value of k to 1 and display days are out of range and Program will not execute further.
11. Now check if the 'N' number of days are in the range or not.
12. If it is in range than calculate date after 'N' days
13. If it is not in the range than change the value of k to 1 and display 'N days are out of range' and Program will not execute further.
14. If days and N days both are in the range than Display Date and Date after N days.
15. Another condition is If the year entered is not a Leap year
16. Than do not increment the value of days number in the array.
17. Print final values
18. End Algorithm

13. disariumNumber

Topics: Functions, Constructors, Object Passing
Algorithm:

1. Start algorithm
2. Declare and initialise integer size to default value. Initialise num to the num entered in main method for parameterised constructor.
3. In function countDigit, calculate the length of the integer num.
4. In recursive function sumOfDigits, Add the digits raised to the power of its position (starting from 0) to a counter.
5. Starting from the right, do n % 10 to get the first digit and raise it to its position (number of digits in the number calculated by another function)
6. Call the function recursively by dividing the integer by 10 (removing the last digit) and decreasing size of number by 1 each time till the size of number is 0
7. In function check, check if the num is equal to the sum of its digits and call the recursive function. Print true or false.

8. End Algorithm

14. eulerFunction

Topics: If Else, Loops
Algorithm:

1. Start algorithm
2. Store first value of double e as 1 according to the formula.
3. Run a while loop with condition flag == false. If the required condition of e2 - e1 < 0.0000001 is true, make flag true.
4. In the while loop, increment count to store the number of iterations taken in the loop. Find the factorial of the count by running for loop from i = 2 till less than equal to count and multiplying in fact. Calculate successive values of e using e = 1/1! + 1/2! + 1/3! .. + 1/n! and store difference between current and previous e value in temp.
5. When condition is met, print the count.
6. End Algorithm.

15. fascinatingNumber

Topics: Loops, Functions, Strings
Algorithm:

1. Start algorithm
2. Take inputs of the lower and upper limit of range and store as int.
3. Declare and initialise count variable to default int value.
4. Start for loop from i = l till less than equal to h with increment of 1.
    1. Check if i is a fascinating number by calling boolean function isFascinating(i)
    2. If Yes, print the number and increment count by 1.
5. isFascinating()
    1. Append the number, number*2 and number*3 in a string str.
    2. Store value of str in a new string tmp and clear str.
    3. Start for loop from i = 0, less than length of tmp with increment of 1.
        1. Inside check if tmp.charAt(i) != 0, if true, append the char to string str.
    4. Make char array using all characters of string str
    5. Sort the digits in ascending alphabetical order.
    6. Check if the string value of the sorted char array is equal to "123456789" and return boolean value of expression.

16. goldbachNumber

Topics: If Else, Loops, Functions
Algorithm:

1. Start algorithm
2. Take number input and check if it is valid by checking if it is within range of 9 to 50, is odd and not negative.
3. Check if the number is an even integer and greater than 4, if true proceed.

4. Check if there are any pairs of two odd integers that add to form the number. Run a nested for loop from [3, num] incrementing by 2 and check if both the iterators (i and j) are prime and i is less than j.
5. In prime checking function, run a for loop from 2 till less than num. If num is divisible by i, increment flag. Check if flag is equal to 0, then the number is prime and return true, otherwise return false.
6. End Algorithm

17. isogramSentence

Topics: Strings, Arrays, Loops, If Else
Algorithm:

1. Start algorithm
2. Take input the sentence and store in String sentence
3. Store length of sentence in int length
4. Run loop from i = 0 till length and seperate the words from the sentence and store them in str[] - in the loop, if sentence[i] is a space, the previous characters form the word. Reset the word variable and move on to the next iteration.
5. Check if the word is an isogram -> Run a for loop from i = 0 till str.length. Check if any character is repeated in the word by running two loops checking every possible comparison of characters in the string.
6. If any comparison of characters in the word is found to be true, increment the flag variable and break. Otherwise continue all iterations. In the end, check flag variable and print final result.
7. End Algorithm.

18. keyboardAnalyser

Topics: Functions, 2D Arrays, Strings
Algorithm:

1. Start algorithm
2. Take input to be analysed
3. In functions findFingerQWERTY and findFingerDVORAK and findFingerCOLEMAK, take input for the character and return the number of the finger that would press that character.
4. In functions findDistanceQWERTY and findFingerDVORAK and findFingerCOLEMAK, * Find the distance to be travelled by each finger to press all keys of the input in succession. If the key is found in the home / middle row of the keyboard, the finger presses the key but only moves if the keys are the middle two keys. Otherwise there is no distance covered. For keys in the top and bottom row, distance moved per keystroke is 1.
5. Run a for loop from i = 0 to less than input.length(). Find distance moved by all fingers and store in an array (there are 8 because thumb fingers are only used for pressing space, none of the keys) and find average of distance moved by each key. The lesser the distance, the more efficient the layout for the given input.
6. Print final output.
7. End Algorithm.

19. luckyNumbers

Topics: Functions, If Else, Loops
Algorithm:

1. Start algorithm
2. Declare instance variables arr[] and int N with default values.
3. Declare and initialise class object
4. Take input and store value of N
5. Start for loop from i = 0 till less than N. Store i+1 in arr[i]. End for loop.
6. Print the array
7. Declare int steps and initialise with numberOfSteps()
8. Start for loop from i = 1 till less than equal to steps. Inside, eliminate every nth number with n starting from 2 and incrementing by 1.
9. Call print()
10. void eliminate()
    1. Start for loop from i = 0 till less than N and increment by 1.
    2. if the element at current index is 0, jump to next iteration.
    3. if the (a+1)th index is divisible by count, change value at that index to 0.
    4. Increment a by 1.
11. int numberOfSteps()
    1. Declare and initialise int steps as 0, i as 2 and N as the calling object's N.
    2. Start a while loop with condition (i <= N)
        1. Declare and initialise int R, short for Remaining, with value N/i
        2. Subtract R from N
        3. Increment i and steps by 1
        4. Close loop
    3. Return steps
12. void print()
    1. Start a for loop to traverse the array
    2. Inside, if (arr[i] != 0), print the element
    3. End for loop
    4. Print a break line

20. potential

Topics: Strings, Loops, Arrays
Algorithm:

1. Start algorithm
2. Declare scanner object and take a string input from user to store the sentence in String sentence
3. Convert sentence to upper case, trim it and append a space to it.
4. Store length of sentence in integer length
5. Declare an integer array called arrSum to store potential of each word and arrStr to store all words of the sentence.
6. Run a for loop from i = 0 till length. Store sentence[i] in char ch. If it is not a space, add the integer value of that character in sum and add the char to word. When a space is detected, all characters of the previous word have been stored in word and potential of that word is stored in sum. Assign word to arrStr[i] and assign sum to arrSum[i]. After every iteration, increment counter by 1.

7. After for loop is complete, use bubble sort to sort the words on basis of the potential of each word. Run a for loop from i = 0 till counter and a nested for loop from j = 0 till less than counter - i - 1. If the potential value of arrSum[j] is greater than the potential value of arrSum[j+1], then swap them and swap the words using temp and tem variables.
8. Print final values
9. End Algorithm.

21. recursiveSortSearch

Topics: Recursions, Functions, Loops, If Else
Algorithm:

1. Start Algorithm
2. Take inputs for the size of array and elements
3. Sort using bubble sort: In ascending Bubble Sort, the adjoining values are compared and exchanged if they are not in order. After one pass, the largest element is put in the end.
    1. if (j == arr.length - i - 1) (If the column index is the last one)
        1. if (i != arr.length - 2) (If the row index is not the second last) is true, then return bubbleSort() incrementing i and reset j to 0.
        2. else, both i and j are at the end of their indexes, print the array (Base Case)
    2. else, compare arr[j] and arr[j+1], swap the values. Then, return bubbleSort() while incrementing j.
4. Search element using binary search. Binary search is the technique which only works in sorted arrays. The search element is compared with the middle element of the array. If the search element matches the middle element, search finishes. If the search element is less than middle, perform binary search in the first half of the array, otherwise perform binary search in the latter part of the array.
    1. Check if the lower index is greater than upper index (Base Case). If true, return -1 (Element not found).
    2. Declare and initialise int m with value (l+u)/2
    3. Start while loop with condition (l <= u)
        1. if (arr[m] == e), return m
        2. if (arr[m] < e), set u = m - 1;
        3. else, set l = m + 1
5. End Algorithm.

22. reverseEncryption

Topics: Strings, Arrays, Loops
Algorithm:

1. Start algorithm
2. Make scanner object and take string input for paragraph.
3. Declare and Initialise a StringTokenizer with inputs as the paragraph and use "." as the delimiter and declare and initialise a 1d array of length of the number of tokens to store the sentences from the paragraph.
4. Separate the sentences from the paragraph and store in a string array using StringTokenizer in a while hasMoreTokens() is true.

5. Run a for loop from i = 0 till sentence.length and add each character in odd sentences with an integer and type cast to character. Reverse the whole sentence of each even sentence.
6. After this encryption, store in file and print.
7. End Algorithm.

23. reverseLinkedList

Topics: Node, LinkedList, Loops, Inheritance
Algorithm:

1. Start algorithm
2. In class Node, declare int data and Node next.
    1. In Nodal constructor, set value of data as d and next node as null.
3. In main(), make object list
    1. Set values of head and next nodes as integer values
    2. Print Values
    3. Reverse List
    4. Print reversed list
4. In Node reverse(),
    1. Initialize three pointers prev as NULL, curr as head and next as NULL.
    2. Iterate through the linked list using while loop. In loop, do following.
        1. Before changing next of current, store next node
           next = curr->next
        2. Now change next of current, This is where actual reversing happens
           curr->next = prev
        3. Move prev and curr one step forward
           prev = curr
           curr = next

24. sentencePalindromeGenerator

Topics: If Else, Loops, Strings, Functions
Algorithm:

1. Start algorithm
2. Take input for sentence using Scanner object and store in String sentence
3. Check if last char of the sentence is not a valid punctuation, exit the program.
4. Declare and initialise words[] using sentence.split(" ")
5. Start for loop iterating through sentence
    1. Declare and initialise boolean b as isPalindrome(words[i])
    2. If it is a palindrome, no changes are required to the word. Append it to the new sentence.
    3. Else, convert the non-palindrome word. To do this, the take a substring of the word excluding the last character.
    4. If the last two characters are repeating, i.e. if the last character of the substring is equal to the last character of the word, then remove another of the last character of the substring.
    5. Now reversing the substring using StringBuffer reverse() function.

6. Append the reverse to the substring
7. Append the substring to the new sentence
8. Exit loop

6. Print the initial and final sentence
7. boolean isPalindrome()
    1. Store the StringBuffer value of the passed string in sb
    2. Reverse sb
    3. Check if the string passed is equal to the string value of sb and return boolean value

25. ticTacToe

Topics: Functions, Strings, 2D Arrays, If Else
Algorithm:

1. Start algorithm
2. Declare a 2D 3x3 char array storing the ticTacToe board and fill it with positions (1, 2, 3... 9)
3. Store two char inputs for characters used by the two players on the ticTacToe board in player1choice and player2choice.
4. Keep going between the turns of players till a tie is reached or one player wins.
5. In function printBoard(), run a for loop for i = 0 to less than 3. Inside, run a nested for loop from j = 0 to less than 3 and print board[i][j]
6. In function determineRowAndColumn(int position), take input for 1d (1, 2, 3...9) position that user wants to place their character at, and store the two dimensional coordinates for that position (like 2, 1 for 8th position and 0,0 for 1st position) in integer variables row and column
7. In function player1move(), make the move on the board for player 1 using their character
8. In function player1move(), make the move on the board for player 2 using their character
9. In function checkWinner(char playerChoice), check if any three adjacent positions in rows, columns or diagonals are occupied by the character representing any player. If found to be true, return true that the player using playerChoice has won!
10. In function checkTie() check if the total number of moves (stored in count) have reached 9. If so, and nobody has won yet, declare a tie and exit function.
11. In functions player1Turn and player2Turn, enter position and store in player1position. Convert the 1d coordinates to index of rows and columns and check if that place has already been occupied on the board. If not, place the player's character on that position on the board. Run a while loop to ensure that if the place is occupied, tell the user to enter a valid position until it is achieved. After a valid move, check if the user has won by calling checkWinner. If a user has won, print and end the game.
12. End Algorithm.

26. towerOfHanoi

Topics: Stack, Functions, If Else
Algorithm:

1. Start Algorithm
2. Main method()
    1. Create Object toh
    2. Create three stacks of size disk to hold the disks - src, dest and aux
    3. Call toh.solution and pass the disks

3. Stack create()
    1. Declare and initialise the stack
    2. Initialise size as size passed
    3. Initialise top as -1
    4. Declare arr with size
    5. Return Stack
4. boolean isFull()
    1. Check and return boolean value if top == size - 1
5. boolean isEmpty()
    1. Check and return boolean value if top == -1
6. void push()
    1. If stack is not full, add an integer to ++top.
7. int pop()
    1. Return the int at index top and decrement top.
8. void moveDisksBetweenPoles()
    1. Push the disk from pole that is not empty - src or dest.
    2. If none of the poles are empty, push from the higher disk to the lower one depending on top disk of both poles.
    3. After pushing the disks, execute moveDisk()
9. void moveDisk()
    1. Print values of disks and rods.
10. void solution()
    1. Calculate the total number of moves required i.e. "pow(2, n) - 1" here n is number of disks.
    2. If number of disks (i.e. n) is even then interchange destination pole and auxiliary pole.
    3. for i = 1 to total number of moves:
       if i%3 == 1:
          legal movement of top disk between source pole and destination pole
       if i%3 == 2:
          legal movement top disk between source pole and auxiliary pole
       if i%3 == 0:
          legal movement top disk between auxiliary pole and destination pole

27. typingTest

Topics: Functions, Strings, Arrays, Exceptions
Algorithm:

1. Declare instance variables:
    1. String input
    2. int WPM, CPM, numberOfWords, correctWords
    3. double start, end, time, accuracy, minutesTaken
2. In function countdown, use thread.sleep to create a '3 2 1' countdown to create anticipation and build atmosphere. Then print the sample input. Put it in a try catch block to handle InterruptedException if the user stops the countdown.

3. In function input, initialise Scanner object. Make a long variable start to store the initial nanoseconds of day. Store input sentences from user and store in variable String input. Make another long variable to store the final nanoseconds of day. Store the difference in nanoseconds (ie the time taken by user to type out the sample input) in time and convert it from nanoseconds to seconds by divididing the difference by 10^9.
4. In function calculate, compute all statistics related to the test.
    1. First of all, the number of words in the sample is 30 and count of characters is 114. Declare a scanner object to separate and store all words entered by user in a String array. Run a for loop from i = 0 to i = bank.length and compare if bank[i] and words[i] are equal. If they are, then increment the integer variable correctWords to store the total number of correct words.
    2. Find total accuracy by dividing and taking percentage of correctWords / numberOfWords and round it upto the second decimal.
    3. Find the WPM (Words per minute) and CPM (Characters per minute) by first converting time taken by user from seconds to minutes, and then dividing correctWords by timeTaken.
5. In function display, print all statistics to user.
6. End Algorithm.

28. verticalBanners

Topics: If Else, Loops, Arrays
Algorithm:

1. Start Algorithm
2. Store value of N in integer n
3. Check if input is invalid
4. Declare and initialise String array teams with size n
5. Declare and set default value to integer highLen
6. Start for loop from i = 0 till less than n with post-increment of 1. Inside loop, take input for word of teams[i]. By the end of the loop, the biggest length from the strings is stored in highLen.
7. Start another for loop from i = 0 till less than highLen with post-increment of 1.
    1. Inside, start another for loop from j = 0 till less than n with post-increment of 1.
    2. Declare and initialise integer len with value as teams[j].length()
    3. If i is greater than equal to len, don't print the letter and rather the tab sequence
    4. Else print the teams[j].charAt(i) character and a tab sequence
    5. Exit inner loop
8. Print a break line
9. Exit outer loop
10. End Algorithm

29. wordleStartChecker

Topics: Arrays, Strings, Functions, If Else
Algorithm:

1. Start algorithm

2. Declare and initialise char[] list with the characters in alphabet in order of frequency of usage in the english language, double freq[] with the frequency percentages of those characters and int nTries as 0.
3. Take input of the count of 5 letter words required and store in nTries
4. Take input of the required number of strings and append them to String word with no whitespaces.
5. Execute function compareWord(), passing the word in lower case
6. Function compareWord()
    1. Declare and Initialise int length as length of string passed
    2. Declare String check and int checkPercent to initial values.
    3. Run for loop from i till less than length, incrementing by 1. Inside, create a word with characters of highest accuracy and store the subsequent frequency percentages.
    4. Declare and initialise int flag and wordPercentage with default values.
    5. Start for loop from i = 0 till less than length with increment of 1.
        1. Check if the ideal word check contains the ith character in string word. If true, increment flag by 1.
        2. wordPercentage

30. wordShift

Topics: Strings, StringTokenizer, Functions
Algorithm:

1. Start algorithm
2. Take input for sentence and store as String, shift as int.
3. Declare and initialise stringtokenizer object with parameters sentence and default delimiter. Store st.countTokens() in int length.
4. Store remainder of shift / length in shift, since shift would only effectively take place less than length times. If input shift is greater than length, shifting process duplicates.
5. Start while loop with condition (st.hasMoreToken()), essentially, traversing through the stringtokenizer
    1. Check if (count < length - shift). If yes, append the nextToken() to s2 and increment count.
    2. Else, append nextToken() to s3
    3. End while loop.
6. Print s3 + s2.

achillesNumber.java

```java
/**
 * An Achilles Number is a number that is powerful but not a perfect power.
 * A Powerful Number is a positive integer N, such that for every prime factor p of N, p2 is also a factor.
 * A Perfect Power is a positive integer N such that it can be expressed as ab, where a and b are natural numbers > 1.
 */

import java.util.LinkedList;
import java.util.Scanner;

public class achillesNumber {
    public static void main(String[] args) {
        // Inputs
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter number: ");
        int num = sc.nextInt();
        sc.close();
        boolean b1 = isPowerful(num);
        boolean b2 = isPerfect(num);
        if (b1 && !b2)
            System.out.println("Number is an achilles number");
        else
            System.out.println("Number is not an achilles number");
    }

    public static boolean isPowerful(int n) {
        int tmp = n;
        // Store all unique prime factors of n
        LinkedList<Integer> lk = new LinkedList<Integer>();
        if (n % 2 == 0) {
            lk.add(2);
            while (n % 2 == 0) {
                n /= 2;
            }
        }
        for (int i = 3; i <= Math.sqrt(n); i += 2) {
            if (n % i == 0) {
                lk.add(i);
                while (n % i == 0) {
                    n /= i;
                }
            }
        }
        if (n > 2)
            lk.add(n);
        // Check if the square of each prime factor is also divisible by n
        int flag = 0;
        for (int i = 0; i < lk.size(); i++) {
            if (tmp % lk.get(i) == 0)
                flag += 1;
        }
        return (flag == lk.size());
    }

    public static boolean isPerfect(int num) {
        for (int i = 2; i < num; i++) {
            for (int j = 2; j < num; j++) {
                int exponent = (int) (Math.pow(i, j));
                if (exponent == num)
                    return true;
            }
        }
        return false;
    }
}
/**
 * Variable        Data        Table
 * num             int         Store user input
 * b1              boolean     Store value from isPowerful()
 * b2              boolean     Store value from isPerfect()
 * flag1, flag2    int         Flag variable to check powerful
 * i, j            int         Counter
 * exponent        int         Used in calculation
 */
```

```
Enter number:
72
Number is an achilles number
Enter number:
36
Number is not an achilles number
Enter number:
108
Number is an achilles number
```

adder.java

```java
// Add time (hours and minutes) by passing arrays using Call by Reference

import java.util.Scanner;

public class adder {
    int[] a;

    adder() {
        a = new int[2];
    }

    public static void main(String[] args) {
        adder X = new adder();
        adder Y = new adder();
        X.readtime();
        Y.readtime();
        adder Z = new adder();
        Z.addtime(X, Y);
        Z.disptime();
    }

    void readtime() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter hours and minutes");
        for (int i = 0; i < a.length; i++)
            a[i] = sc.nextInt();
        sc.close();
    }

    // Both arrays are passed as objects and final values are stored in calling object
    void addtime(adder X, adder Y) {
        this.a[0] = X.a[0] + Y.a[0];
        this.a[1] = X.a[1] + Y.a[1];
        if (this.a[1] > 60) {
            this.a[0] += 1;
            this.a[1] %= 60;
        }
    }

    void disptime() {
        System.out.println("hours = " + a[0] + " and minutes = " + a[1]);
    }
}
/**
 * Variable        Data        Table
 * a               int[]       Instance variable array
 * i               int         Used as count
 */
```

```
Enter hours and minutes
13
45
Enter hours and minutes
7
10
hours = 20 and minutes = 55
Enter hours and minutes
8
22
Enter hours and minutes
4
55
hours = 13 and minutes = 17
Enter hours and minutes
14
20
Enter hours and minutes
19
42
hours = 34 and minutes = 2
```

arrShiftSortMerge.java

```java
// You are given two integer arrays nums1 and nums2, sorted in non-decreasing order, and two integers m and n,
// representing the number of elements in nums1 and nums2 respectively.
// Merge nums1 and nums2 into a single array sorted in non-decreasing order.
// The final sorted array should not be returned by the function, but instead be stored inside the array nums1.
// To accommodate this, nums1 has a length of m + n, where the first m elements denote the elements that should
// be merged, and the last n elements are set to 0 and should be ignored. nums2 has a length of n.
import java.util.Scanner;
public class arrShiftSortMerge {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter size of arr1");
        int n1 = sc.nextInt();
        System.out.println("Enter " + n1 + " elements");
        int[] arr1 = new int[n1];
        for (int i = 0; i < n1; i++)
            arr1[i] = sc.nextInt();
        System.out.println("Enter size of arr2");
        int n2 = sc.nextInt();
        System.out.println("Enter " + n2 + " elements");
        int[] arr2 = new int[n2];
        for (int i = 0; i < n2; i++)
            arr2[i] = sc.nextInt();
        // Create new array tmp with size n1 + n2 and filled with elements of arr1 and zeroes
        int[] tmp = new int[n1+n2];
        for(int i = 0; i < n1+n2; i++) {
            if (i < n1)
                tmp[i] = arr1[i];
            else
                tmp[i] = 0;
        }
        merge(tmp, n1, arr2, n2);
    }

    public static void merge(int[] arr1, int n1, int[] arr2, int n2) {
        int i = n1 - 1;
        int j = n2 - 1;
        int k = n1 + n2 - 1;
        while (j >= 0) {
            if (i >= 0 && arr1[i] > arr2[j])
                arr1[k--] = arr1[i--];
            else
                arr1[k--] = arr2[j--];
        }
        for (i = 0; i < arr1.length; i++)
            System.out.print(arr1[i] + " ");
    }
}
```

```
Enter size of arr1
4
Enter 4 elements
1
3
4
5
Enter size of arr2
4
Enter 4 elements
2
4
6
8
1 2 3 4 4 5 6 8 Enter size of arr1
3
Enter 3 elements
5
8
9
Enter size of arr2
3
Enter 3 elements
4
7
8
4 5 7 8 8 9
```

arrayTools.java

```java
import java.util.Arrays;
import java.util.Scanner;

public class arrayTools {
    static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {
        int[][] arr = input();
        print(arr);
        transpose(arr);
        rotate(arr);
        System.out.println("Enter number to multiply with");
        multiply1(arr, sc.nextInt());
    }

    static int[][] input() {
        System.out.println("Enter number of rows: ");
        int r = sc.nextInt();
        System.out.println("Enter number of columns: ");
        int c = sc.nextInt();
        System.out.println("Enter " + (r * c) + " elements");
        int[][] arr = new int[r][c];
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                arr[i][j] = sc.nextInt();
            }
        }
        return arr;
    }

    static void print(int[][] arr) {
        for (int[] ints : arr) {
            System.out.println(Arrays.toString(ints));
        }
    }

    static void transpose(int[][] arr) {
        int r = arr.length;
        int c = arr[0].length;
        int[][] trans = new int[c][r];
        for (int i = 0; i < c; i++) {
            for (int j = 0; j < r; j++) {
                trans[i][j] = arr[j][i];
            }
        }
        System.out.println("Transposed array: ");
        print(trans);
    }

    static void rotate(int[][] arr) {
        int[][] rot = new int[arr.length][arr[0].length];
        for (int i = 0, k = 2; i < rot.length; i++, k--) {
            for (int j = 0; j < rot[i].length; j++) {
                rot[j][k] = arr[i][j];
            }
        }
        System.out.println("pi/2 rad rotated clockwise matrix: ");
        print(rot);
    }

    static void multiply1(int[][] a, int b) {
        for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < a[i].length; j++) {
                a[i][j] *= 2;
            }
        }
        System.out.println("Product matrix: ");
        print(a);
    }

    static void addSubtract(int[][] a, int[][] b) {
        if (a.length != b.length && a[0].length != b[0].length) {
            System.out.println("Matrices have different sizes!");
            return;
        }
        int[][] sum = new int[a.length][a[0].length];
        int[][] difference = new int[a.length][a[0].length];
        for (int i = 0; i < sum.length; i++) {
            for (int j = 0; j < sum[i].length; j++) {
                sum[i][j] = a[i][j] + b[i][j];
```

```
                difference[i][j] = a[i][j] - b[i][j];
            }
        }
        System.out.println("Sum of the matrices: ");
        print(sum);
        System.out.println("Difference between the matrices: ");
        print(difference);
    }

    static void arraySums(int[][] arr, int r, int c) {
        System.out.println("\nSum of left diagonals\n");
        int dL = 0, dR = 0;
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                if (i == j) {
                    System.out.print(arr[i][j] + " ");
                    dL += arr[i][j];
                }
            }
        }
        System.out.println(dL);
        System.out.println("\nSum of right diagonals\n");
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                if (i + j == (r - 1)) {
                    System.out.print(arr[i][j] + " ");
                    dR += arr[i][j];
                }
            }
        }
        System.out.println(dR);
    }

    static void rowShift(int[][] arr, int n) {
        int tmp;
        for (int k = 1; k <= n; k++) {
            for (int i = 0; i < arr.length; i++) {
                for (int j = 0; j < arr[i].length; j++) {
                    tmp = arr[0][j];
                    arr[0][j] = arr[i][j];
                    arr[i][j] = tmp;
                }
            }
        }
    }
}
```

```
Enter number of rows:
3
Enter number of columns:
3
Enter 9 elements
1
2
3
4
5
6
7
8
9
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
Transposed array:
[1, 4, 7]
[2, 5, 8]
[3, 6, 9]
pi/2 rad rotated clockwise matrix:
[7, 4, 1]
[8, 5, 2]
[9, 6, 3]
Enter number to multiply with
3
Product matrix:
[2, 4, 6]
[8, 10, 12]
[14, 16, 18]
```

binaryDecimalShenanigans.java

```java
import java.util.Scanner;

public class binaryDecimalShenanigans {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter decimal number: ");
        int decimal = sc.nextInt();
        System.out.print("Enter binary number: ");
        int binary = sc.nextInt();
        sc.close();
        System.out.println("Binary to Decimal: " + binaryToDecimal(binary));
        System.out.println("Binary to Decimal using recursion: " + binaryToDecimalRecursion(binary));
        System.out.println("Decimal to Binary: " + decimalToBinary(decimal));
        System.out.println("Decimal to Binary using Recursion: " + decimalToBinaryRecursion(decimal));
    }

    static int binaryToDecimal(int n) {
        String binary = String.valueOf(new StringBuffer("" + n).reverse());
        int decimal = 0;
        for (int i = 0; i < binary.length(); i++) {
            if (binary.charAt(i) == '1') {
                decimal += (int) (Math.pow(2, i));
            }
        }
        return decimal;
    }

    static int decimalToBinary(int decimal) {
        String binary = "";
        while (decimal > 0) {
            binary += decimal % 2;
            decimal /= 2;
        }
        return Integer.parseInt(new StringBuffer(binary).reverse().toString());
    }

    static int decimalToBinaryRecursion(int n) {
        if (n == 0)
            return 0;
        return n % 2 + 10 * decimalToBinaryRecursion(n / 2);
    }

    static int binaryToDecimalRecursion(int n) {
        if (n == 0)
            return 0;
        return n % 10 + 2 * binaryToDecimalRecursion(n / 10);
    }
}
```

```
Enter decimal number: 42
Enter binary number: 101010
Binary to Decimal: 42
Binary to Decimal using recursion: 42
Decimal to Binary: 101010
Decimal to Binary using Recursion: 101010
Enter decimal number: 27
Enter binary number: 11011
Binary to Decimal: 27
Binary to Decimal using recursion: 27
Decimal to Binary: 11011
Decimal to Binary using Recursion: 11011
Enter decimal number: 8
Enter binary number: 1000
Binary to Decimal: 8
Binary to Decimal using recursion: 8
Decimal to Binary: 1000
Decimal to Binary using Recursion: 1000
```

blinsMaker.java

```java
// Find how many 'blins' (pancakes) you can make yourself when babushka (grandma) isn't home

import java.util.Scanner;

public class blinsMaker {
    public static void main(String[] args) {
        int eggs;
        int milk; // millilitre
        int flour; // grams
        // Recipe for single blin
        int minEggs = 2;
        int minMilk = 200;
        int minFlour = 100;
        // Taking inputs
        Scanner sc = new Scanner(System.in);
        System.out.println("How many eggs you have?");
        eggs = sc.nextInt();
        System.out.println("How much milk you have?");
        milk = sc.nextInt();
        System.out.println("How much flour you have?");
        flour = sc.nextInt();
        sc.close();
        // Calculating if there is enough for one meal
        if (eggs < minEggs || milk < minMilk || flour < minFlour) // One meal is four blins
            System.out.println("There is not enough for a single meal :(");
        else { // Calculate Minimum possible portions
            int eggPortions = eggs / minEggs;
            int milkPortions = milk / minMilk;
            int flourPortions = flour / minFlour;
            int smallest;
            if ((eggPortions < milkPortions && eggPortions < flourPortions)) {
                smallest = eggPortions;
            } else {
                if (milkPortions < eggPortions && milkPortions < flourPortions) smallest = milkPortions;
                else smallest = flourPortions;
            }
            System.out.println("You can make " + smallest * 4 + " blins, or " + smallest + " portions");
            System.out.println("You need " + smallest * minEggs + " eggs and " +
                    (eggs - (smallest * minEggs)) + " will not be used");
            System.out.println("You need " + smallest * minMilk + "ml milk and " +
                    (milk - (smallest * minMilk)) + " will not be used");
            System.out.println("You need " + smallest * minFlour + "g flour and " +
                    (flour - (smallest * minFlour)) + " will not be used");
        }
        System.out.println("Blinmaker shutting down...");
    }
}
/**
 * Variable            Data        Table
 * eggs, milk, flour   int         Store amount of food user has
 * min""               int         (constant) Store minimum amount required for 1 pancake
 * portions            int         (constant) Minimum portions required
 * smallest            int         Store smallest out of eggs, milk and flour
 */
```

How many eggs you have?
6
How much milk you have?
100
How much flour you have?
150
There is not enough for a single meal :(
Blinmaker shutting down...
How many eggs you have?
12
How much milk you have?
300
How much flour you have?
400
You can make 4 blins, or 1 portions
You need 2 eggs and 10 will not be used
You need 200ml milk and 100 will not be used
You need 100g flour and 300 will not be used
Blinmaker shutting down...
How many eggs you have?
24
How much milk you have?
3000
How much flour you have?
50000
You can make 48 blins, or 12 portions
You need 24 eggs and 0 will not be used
You need 2400ml milk and 600 will not be used
You need 1200g flour and 48800 will not be used
Blinmaker shutting down...

boundarySort.java

```java
// Sort only boundary elements of a 2d array

import java.util.Arrays;
import java.util.Scanner;

public class boundarySort {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter number of rows: ");
        int r = sc.nextInt();
        System.out.println("Enter number of coln: ");
        int c = sc.nextInt();
        System.out.println("Enter " + (r * c) + " elements: ");
        int[][] arr = new int[r][c];
        // Store boundary elements of arr in a new 1d array
        int[] boundary = new int[(r * c) - (r - 2) * (c - 2)];
        int count = 0;
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                arr[i][j] = sc.nextInt();
                if (i == 0 || j == 0 || i == (r - 1) || j == (c - 1)) {
                    boundary[count] = arr[i][j];
                    ++count;
                }
            }
        }
        sc.close();
        // Sort the 1d array using selection sort
        int min, temp;
        for (int i = 0; i < boundary.length; i++) {
            min = i;
            for (int j = i + 1; j < boundary.length; j++) {
                if (boundary[j] < boundary[min])
                    min = j;
            }
            temp = boundary[i];
            boundary[i] = boundary[min];
            boundary[min] = temp;
        }
        // Put the sorted elements back in 2d array
        count = 0;
        for (int i = 0; i < c; i++) {
            arr[0][i] = boundary[count];
            ++count;
        }
        for (int i = 1; i < r; i++) {
            arr[i][c - 1] = boundary[count];
            ++count;
        }
        for (int i = c - 2; i >= 0; i--) {
            arr[r - 1][i] = boundary[count];
            ++count;
        }
        for (int i = r - 2; i >= 1; i--) {
            arr[i][0] = boundary[count];
            ++count;
        }
        for (int i = 0; i < r; i++) {
            System.out.println(Arrays.toString(arr[i]));
        }
    }
}
/**
 * Variable        Data        Table
 * r, c            int         Store rows and columns for 2d array
 * arr             int[][]     Store 2D array
 * boundary        int[]       Extract and store boundary elements in 1d array
 * count, i, j     int         Iterators
 * min, temp       int         Used in sorting
 */
```

```
Enter number of rows:
3
Enter number of coln:
3
Enter 9 elements:
1
3
2
4
6
5
7
9
8
[1, 2, 3]
[9, 6, 4]
[8, 7, 5]
Enter number of rows:
4
Enter number of coln:
4
Enter 16 elements:
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
[1, 2, 3, 4]
[16, 6, 7, 5]
[15, 10, 11, 8]
[14, 13, 12, 9]
```

camelConc.java

```java
// To concatenate a sentence using camel concatenation

import java.util.Scanner;

public class camelConc {
    public static void main(String[] args) {
        Scanner Sc = new Scanner(System.in);
        System.out.print("Enter sentence: ");
        String sentence = Sc.nextLine();
        sentence = sentence.trim();
        sentence += " ";
        String[] str = new String[sentence.length()];
        int counter = 0;
        int a = 0;
        char ch;
        boolean b;
        String word = "";
        // Separating words and putting in string array
        for (int i = 0; i < sentence.length(); i++) {
            ch = sentence.charAt(i);
            b = Character.isWhitespace(ch);
            if (b == true) {
                word = sentence.substring(a, i);
                str[counter] = word;
                a = i + 1;
                counter += 1;
                word = "";
            }
        }
        // Concatenating the words
        String finalStr = "";
        finalStr += str[0].toLowerCase();
        for (int i = 1; i < counter; i++) {
            word = (str[i].substring(0, 1).toUpperCase()) + str[i].substring(1).toLowerCase();
            finalStr += word;
        }
        System.out.println("The final word is: " + finalStr);
        Sc.close();
    }
}
/**
 * Variable data table
 * Variable     Type    Function
 * sentence     String  To store sentence
 * str[]        String  To store words of sentence in array
 * counter      int     Used as counter for array
 * an            int     Used to separate words
 * word         String  Used to separate words
 * finalStr     String  Used to store output
 */
```

```
Enter sentence: superior variable
The final word is: superiorVariable
Enter sentence: better than pascal
The final word is: betterThanPascal
Enter sentence: the true way of monke
The final word is: theTrueWayOfMonke
```

chess.java

```java
/**
 * An input of 2 index positions in a 2d Array of size 8x8 will be taken. The program
 * should then check which chess piece can move from the first index to the second in 1 move.
 * The chess pieces move as follows:
 * King - 1 space in any direction
 * Queen - any number of spaces in forward, backward, left, right and diagonal directions
 * Bishop - any number of spaces in diagonal directions
 * Knight - 2 spaces in forward, backward, left or right directions followed by 1 space perpendicular to it.
 * Rook - any number of spaces in forward, backward, left and right directions
 * Pawn - 1 space in the forward direction only.
 */

import java.util.Scanner;

public class chess {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter start indexes");
        int x1 = sc.nextInt();
        int y1 = sc.nextInt();
        System.out.println("Enter end indexes");
        int x2 = sc.nextInt();
        int y2 = sc.nextInt();
        int flag = 0;
        sc.close();
        // Program is easily solved using distance formula between two points
        // The distance formula in geometry is ((y2-y1)^2 + (x2-x1)^2)^0.5
        // The right distance needed to attack a piece is constant so just needs to be checked once.
        double distance = Math.sqrt(Math.pow((y2 - y1), 2) + Math.pow((x2 - x1), 2));
        if (distance == 1 || distance == Math.sqrt(2)) {
            System.out.println("King");
            flag = 1;
        }
        if (distance % Math.sqrt(2) == 0) {
            System.out.println("Bishop");
            flag = 1;
        }
        if (x1 == x2 || y1 == y2) {
            System.out.println("Rook");
            flag = 1;
        }
        if (distance == Math.sqrt(5)) {
            System.out.println("Horsey!");
        }
        if (y1 == y2 && (x2 + 1 == x1)) {
            System.out.println("Pawn");
            flag = 1;
        }
        if (flag == 1) {
            System.out.println("Queen");
        }
    }
}
/**
 * Variable        Data        Table
 * x1, y1          int         Store coordinates of first point on chess board
 * x2, y2          int         Store coordinates of second point on chess board
 * distance        int         Calculate and store distance between the points
 * flag            int         Check for queen
 */
```

```
Enter start indexes
0
1
Enter end indexes
4
5
Bishop
Queen
Enter start indexes
2
2
Enter end indexes
3
3
King
Bishop
Queen
Enter start indexes
0
0
Enter end indexes
4
0
Rook
Queen
Enter start indexes
0
0
Enter end indexes
2
1
Horsey!
```

circularQueue.java

```java
import java.util.*;

public class circularQueue {
    int size;
    int[] arr;
    int r, f;
    int c;

    circularQueue(int size) {
        f = -1;
        r = -1;
        c = 0;
        this.size = size;
        arr = new int[this.size];
    }

    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter max size: ");
        circularQueue cq = new circularQueue(sc.nextInt());
        System.out.println("Enter number of elements you want to enter: ");
        cq.c = sc.nextInt();
        for (int i = 0; i < cq.c; i++) {
            cq.arr[i] = sc.nextInt();
        }
        System.out.println(Arrays.toString(cq.arr));
        System.out.println("Popping one element");
        System.out.println(cq.pop());
        System.out.println("Enter element to push: ");
        int e = sc.nextInt();
        sc.close();
        cq.push(e);
        System.out.println(Arrays.toString(cq.arr));
    }

    int pop() {
        if (isEmpty())
            System.out.println(Integer.MIN_VALUE);
        f %= c;
        return arr[++f];
    }

    void push(int n) {
        if (isFull())
            System.out.println(Integer.MAX_VALUE);
        r = (r + 1) % arr.length;
        arr[r] = n;
        c++;
        if (f == -1)
            f = r;
    }

    boolean isEmpty() {
        return (c == 0);
    }

    boolean isFull() {
        return (c == arr.length);
    }
}
```

```
Enter max size:
10
Enter number of elements you want to enter:
5
2
3
4
5
6
[2, 3, 4, 5, 6, 0, 0, 0, 0, 0]
Popping one element
2
Enter element to push:
7
[7, 3, 4, 5, 6, 0, 0, 0, 0, 0]
Enter max size:
5
Enter number of elements you want to enter:
5
5
4
3
2
1
[5, 4, 3, 2, 1]
Popping one element
5
Enter element to push:
3
2147483647
[3, 4, 3, 2, 1]
Enter max size:
10
Enter number of elements you want to enter:
6
9

2
4
5
6
7
```

```
[9, 2, 4, 5, 6, 7, 0, 0, 0, 0]
Popping one element
9
Enter element to push:
0
[0, 2, 4, 5, 6, 7, 0, 0, 0, 0]
```

concentricNumbers.java

```java
/**
 * Given a positive integer n, print the matrix filled with rectangle pattern as shown below:
 * a a a a a
 * a b b b a
 * a b c b a
 * a b b b a
 * a a a a a
 * where a = n, b = n – 1,c = n – 2 and so on.
 */

import java.util.Scanner;

class concentricNumbers {
    public static void main(String[] args) {
        // Taking inputs
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter no");
        int n = sc.nextInt();
        sc.close();
        // This program is done using the distance formula
        n = 2 * n - 1;
        int m = (n - 1) / 2;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(Math.max(Math.abs(i - m), Math.abs(j - m)) + 1 + " ");
            }
            System.out.println();
        }
    }
}
/**
 * Variable        Data        Table
 * n               int         Store number entered by user
 * n, m            int         Used for calculation
 * i, j            int         Iterators
 */
```

```
Enter no
2
2 2 2
2 1 2
2 2 2
Enter no
3
3 3 3 3 3
3 2 2 2 3
3 2 1 2 3
3 2 2 2 3
3 3 3 3 3
Enter no
4
4 4 4 4 4 4 4
4 3 3 3 3 3 4
4 3 2 2 2 3 4
4 3 2 1 2 3 4
4 3 2 2 2 3 4
4 3 3 3 3 3 4
4 4 4 4 4 4 4
```

dateCalculator.java

```java
import java.util.Scanner;

public class dateCalculator {
    public static boolean isLeapYear(int y) {
        boolean ret = false;

        if (y % 400 == 0) {
            ret = true;
        } else if (y % 100 == 0) {
            ret = false;
        } else ret = y % 4 == 0;

        return ret;
    }

    public static String computeDate(int day, int year) {
        int[] monthDays = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
        String[] monthNames = {"JANUARY", "FEBRUARY", "MARCH", "APRIL", "MAY",
                "JUNE", "JULY", "AUGUST", "SEPTEMBER", "OCTOBER", "NOVEMBER", "DECEMBER"};

        boolean leap = isLeapYear(year);

        if (leap) {
            monthDays[1] = 29;
        }

        int i = 0;
        int daySum = 0;
        for (i = 0; i < monthDays.length; i++) {
            daySum += monthDays[i];
            if (daySum >= day) {
                break;
            }
        }

        int date = day + monthDays[i] - daySum;

        StringBuffer sb = new StringBuffer();
        sb.append(date);
        sb.append("TH ");
        sb.append(monthNames[i]);
        sb.append(", ");
        sb.append(year);

        return sb.toString();
    }

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("DAY NUMBER: ");
        int dayNum = in.nextInt();
        System.out.print("YEAR: ");
        int year = in.nextInt();
        System.out.print("DATE AFTER (N DAYS): ");
        int n = in.nextInt();
        in.close();
        if (dayNum < 1 || dayNum > 366) {
            System.out.println("DAY NUMBER OUT OF RANGE");
            return;
        }

        if (n < 1 || n > 100) {
            System.out.println("DATE AFTER (N DAYS) OUT OF RANGE");
            return;
        }

        String dateStr = computeDate(dayNum, year);

        int nDays = dayNum + n;
        int nYear = year;
        boolean leap = isLeapYear(year);

        if (leap && nDays > 366) {
            nYear = nYear + 1;
            nDays = nDays - 366;
        } else if (nDays > 365) {
            nYear = nYear + 1;
            nDays = nDays - 365;
        }

```

```
81          String nDateStr = computeDate(nDays, nYear);
82
83          System.out.println();
84          System.out.println("DATE: " + dateStr);
85          System.out.println("DATE AFTER " + n + " DAYS: " + nDateStr);
86      }
87  }
```

DAY NUMBER: 14
YEAR: 2004
DATE AFTER (N DAYS): 50

DATE: 14TH JANUARY, 2004
DATE AFTER 50 DAYS: 4TH MARCH, 2004
DAY NUMBER: 231
YEAR: 1978
DATE AFTER (N DAYS): 31

DATE: 19TH AUGUST, 1978
DATE AFTER 31 DAYS: 19TH SEPTEMBER, 1978
DAY NUMBER: 200
YEAR: 2556
DATE AFTER (N DAYS): 140
DATE AFTER (N DAYS) OUT OF RANGE

disariumNumber.java

```java
public class disariumNumber {
    int num;
    int size;

    disariumNumber(int nn) {
        this.num = nn;
        this.size = 0;
    }

    public static void main(String[] args) {
        disariumNumber o = new disariumNumber(135);
        o.countDigit();
        o.check();
        System.out.println(o.sumOfDigits(135, 3));

        disariumNumber o2 = new disariumNumber(89);
        o2.countDigit();
        o2.check();
        System.out.println(o2.sumOfDigits(89, 2));

        disariumNumber o3 = new disariumNumber(80);
        o3.countDigit();
        o3.check();
        System.out.println(o3.sumOfDigits(80, 2));
    }

    void countDigit() {
        this.size = String.valueOf(this.num).length();
    }

    int sumOfDigits(int n, int p) {
        return p == 0 ? 0 : (int) Math.pow(n % 10, p) +
                this.sumOfDigits(n / 10, p - 1);
    }

    void check() {
        if (this.num == this.sumOfDigits(this.num, this.size)) {
            System.out.println("true");
        } else {
            System.out.println("false");
        }
    }
}
```

true
135
true
135
true
89
false
8
true
135
true
89
false
8

eulerFunction.java

```java
// The Euler number e is used as the base of natural logarithm
// It can be approximated using the formula
// e = 1/1! + 1/2! + 1/3! .. + 1/n!
// Write a program in Java that approximates e using a loop that terminates when the
// difference between two successive values of e differ by less than 0.0000001
public class eulerFunction {
    public static void main(String[] args) {
        double e = 1.0d;
        int count = 1;
        boolean flag = false;
        while (flag == false) {
            ++count;
            // find fact of count
            int fact = 1;
            for (int i = 2; i <= count; i++) {
                fact *= i;
            }
            double temp = e + (1.0 / fact);
            if (temp - e < 0.0000001) {
                System.out.println(count);
                break;
            }
            e = temp;
        }
    }
}
```

11

fascinatingNumber.java

```java
// A Fascinating number is one which when multiplied by 2 and 3 and then, after the results are concatenated
// with the original number, the new number contains all the digits from 1 to 9 exactly once.
// There can be any number of zeroes and are to be ignored.
// Accept two positive integers m and n, where m must be less than n and the values of both m and n
// must be greater than 99 and less than 10000 as user input. Display all Fascinating numbers that are
// in the range between m and n (both inclusive) and output them along with the frequency

import java.util.Arrays;
import java.util.Scanner;

public class fascinatingNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter lower range: ");
        int l = sc.nextInt();
        System.out.println("Enter higher range: ");
        int h = sc.nextInt();
        sc.close();
        int count = 0;
        for (int i = l; i <= h; i++) {
            if (isFascinating(i)) {
                System.out.print(i + " ");
                count += 1;
            }
        }
        System.out.println("\nNo of fascinating numbers found: " + count);
    }

    static boolean isFascinating(int no) {
        int two = no * 2;
        int three = no * 3;
        String str = "" + no + two + three;
        // Removing all zeroes from number
        String tmp = str;
        str = "";
        for (int i = 0; i < tmp.length(); i++) {
            char ch = tmp.charAt(i);
            if (ch != '0')
                str += ch;
        }
        // Sorting digits in ascending order
        char[] digits = str.toCharArray();
        Arrays.sort(digits);
        String sortedDigits = String.valueOf(digits);
        return sortedDigits.equals("123456789");
    }
}
```

```
Enter lower range:
1
Enter higher range:
1000
192 219 273 327
No of fascinating numbers found: 4
Enter lower range:
1000
Enter higher range:
5000
1902 1920 2019 2190 2703 2730 3027 3270
No of fascinating numbers found: 8
```

goldbachNumber.java

```java
/*A Goldbach number is a positive even integer that can be expressed as the sum of two odd primes.
Note: All even integer numbers greater than 4 are Goldbach numbers.
Example:
6 = 3 + 3
10 = 3 + 7
10 = 5 + 5
Hence, 6 has one odd prime pair 3 and 3. Similarly, 10 has two odd prime pairs, i.e. 3, 7 and 5, 5.
Write a program to accept an even integer N where N > 9 and N < 50. Find all the odd prime pairs whose sum is equal to the number N.
*/

import java.util.Scanner;

public class goldbachNumber {
    public static void main(String[] args) {
        // Taking inputs
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter number");
        int num = sc.nextInt();
        sc.close();
        if (num < 0 || num % 2 != 0 || num <= 9 || num >= 50) { // Checking if number is invalid
            System.out.println("Invalid input");
            System.exit(0);
        }
        if (num % 2 == 0 && num > 4) { // If num is even and greater than 4, then its goldback
            // Finding pairs
            System.out.println("It is a goldbach number \nPairs are:");
            for (int i = 3; i < num; i += 2) {
                for (int j = 3; j < num; j += 2) {
                    if (isPrime(i) && isPrime(j) && i + j == num && i <= j) { // If both numbers are odd primes and they add to the original no, they are required pairs
                        System.out.println(i + " " + j); // Printing the pairs
                    }
                }
            }
        } else {
            System.out.println("It's not a goldbach number");
        }
    }

    public static boolean isPrime(int num) { // Checking if number is a prime
        int flag = 0;
        for (int i = 2; i < num; i++) {
            if (num % i == 0) {
                flag += 1;
                break;
            }
        }
        return flag == 0;
    }
}
```

```
Enter number
14
It is a goldbach number
Pairs are:
3 11
7 7
Enter number
28
It is a goldbach number
Pairs are:
5 23
11 17
Enter number
12
It is a goldbach number
Pairs are:
5 7
```

isogramSentence.java

```java
import java.util.Scanner;

public class isogramSentence {
    public static void main(String[] args) {
        Scanner Sc = new Scanner(System.in);
        System.out.println("Enter sentence: ");
        String sentence = Sc.nextLine();
        Sc.close();
        sentence += " ";
        int length = sentence.length();
        String word = "";
        char h;
        String[] str = new String[length];
        int count = 0;
        // Seperating words of the sentence and putting in string array
        for (int i = 0; i < length; i++) {
            h = sentence.charAt(i);
            if (h != 32)
                word += h;
            if (h == 32) {
                str[count] = word;
                count += 1;
                word = "";
            }
        }
        // Checking if each word is an isogram
        for (int x = 0; x < count; x++) {
            int a = str[x].length();
            int flag = 0;
            char ch, as;
            for (int i = 0; i < (a - 1); i++) {
                ch = str[x].charAt(i);
                for (int j = (i + 1); j < a; j++) {
                    as = str[x].charAt(j);
                    if (ch == as) {
                        flag += 1;
                        break;
                    }
                }
                if (flag > 0) {
                    System.out.println(str[x] + " is not an isogram");
                    break;
                }
            }
            if (flag == 0)
                System.out.println(str[x] + " is an isogram");
        }
    }
}
/**
 * Variable Data Table
 * Variable     Data    Table
 * sentence     String  To store sentence
 * length       int     To store length of sentence
 * word         String  Used in seperating words
 * h            char    Used in seperating words
 * str[]        String  Used in storing words
 * count        int     Used to store number of words
 * a            int     Used to check if word is isogram
 * flag         int     Used to check if word is isogram
 * ch, as       char    Used to check if word is isogram
 */
```

Enter sentence:
some words might not be an isogram
some is an isogram
words is an isogram
might is an isogram
not is an isogram
be is an isogram
an is an isogram
isogram is an isogram
Enter sentence:
but that is a sacrifice
but is an isogram
that is not an isogram
is is an isogram
a is an isogram
sacrifice is not an isogram
Enter sentence:
i am willing to take
i is an isogram
am is an isogram
willing is not an isogram
to is an isogram
take is an isogram

keyboardAnalyser.java

```java
// Comparing the efficiency of various keyboard layouts
// Based on user input, the amount of distance that needs to be travelled to press each key is calculated, provided all fingers are used
// The layouts in which less distance is travelled is most efficient.
// Currently there are qwerty, dvorak and colemak layouts which are compared.

import java.util.Scanner;

public class keyboardAnalyser {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter sentence: ");
        String sentence = sc.nextLine();
        sc.close();
        sentence = sentence.trim().concat(" ");
        System.out.println("Distance moved by fingers to type above sentence: ");
        System.out.println("\nQwerty: ");
        wordFingerDistance(sentence, "QWERTY");
        System.out.println("\nDvorak: ");
        wordDistance(sentence, "DVORAK");
        wordFingerDistance(sentence, "DVORAK");
        System.out.println("\nColemak: ");
        wordDistance(sentence, "COLEMAK");
        wordFingerDistance(sentence, "COLEMAK");
    }

    // Finding the finger required to press a given letter for all layouts
    static int findFingerQWERTY(char a) {
        if ("qaz".contains(Character.toString(a))) return 0;
        if ("wsx".contains(Character.toString(a))) return 1;
        if ("edc".contains(Character.toString(a))) return 2;
        if ("rfvtgb".contains(Character.toString(a))) return 3;
        if ("yhnujm".contains(Character.toString(a))) return 4;
        if ("ik".contains(Character.toString(a))) return 5;
        if ("ol".contains(Character.toString(a))) return 6;
        if ("p".contains(Character.toString(a))) return 7;
        else return 8;
    }

    static int findFingerDVORAK(char a) {
        if ("a".contains(Character.toString(a))) return 0;
        if ("oq".contains(Character.toString(a))) return 1;
        if ("ej".contains(Character.toString(a))) return 2;
        if ("pukyix".contains(Character.toString(a))) return 3;
        if ("fdbghm".contains(Character.toString(a))) return 4;
        if ("ctw".contains(Character.toString(a))) return 5;
        if ("rnv".contains(Character.toString(a))) return 6;
        if ("lsz".contains(Character.toString(a))) return 7;
        else return 8;
    }

    static int findFingerCOLEMAK(char a) {
        if ("qaz".contains(Character.toString(a))) return 0;
        if ("wrx".contains(Character.toString(a))) return 1;
        if ("fsc".contains(Character.toString(a))) return 2;
        if ("ptvgdb".contains(Character.toString(a))) return 3;
        if ("jhklne".contains(Character.toString(a))) return 4;
        if ("ue".contains(Character.toString(a))) return 5;
        if ("yi".contains(Character.toString(a))) return 6;
        if ("o".contains(Character.toString(a))) return 7;
        else return 8;
    }

    // Finding if the character is on the home row (finger doesn't to move to press it) or not on the home row (the finger has to move to press it)
    static int findDistanceQWERTY(int finger, int ch) {
        int distance = 0;
        if (finger == 0) {
            if (ch != 'a')
                distance += 1;
        }
        if (finger == 1) {
            if (ch != 's')
                distance += 1;
        }
        if (finger == 2) {
            if (ch != 'd')
                distance += 1;
        }
        if (finger == 3) {
            if (ch != 'f')
                distance += 1;
        }
        if (finger == 4) {
            if (ch != 'j') {
                distance += 1;
            }
        }
        if (finger == 5) {
            if (ch != 'k') {
                distance += 1;
            }
        }
        if (finger == 6) {
            if (ch != 'l') {
                distance += 1;
            }
        }
        if (finger == 7) {
```

```
            distance += 1;
        }
        return distance;
    }

    static int findDistanceDVORAK(int finger, int ch) {
        int distance = 0;
        if (finger == 1) {
            if (ch != 'o')
                distance += 1;
        }
        if (finger == 2) {
            if (ch != 'e')
                distance += 1;
        }
        if (finger == 3) {
            if (ch != 'u')
                distance += 1;
        }
        if (finger == 4) {
            if (ch != 'h') {
                distance += 1;
            }
        }
        if (finger == 5) {
            if (ch != 't') {
                distance += 1;
            }
        }
        if (finger == 6) {
            if (ch != 'n') {
                distance += 1;
            }
        }
        if (finger == 7) {
            if (ch != 's') {
                distance += 1;
            }
        }
        return distance;
    }

    static int findDistanceCOLEMAK(int finger, int ch) {
        int distance = 0;
        if (finger == 0) {
            if (ch != 'a')
                distance += 1;
        }
        if (finger == 1) {
            if (ch != 'r')
                distance += 1;
        }
        if (finger == 2) {
            if (ch != 's')
                distance += 1;
        }
        if (finger == 3) {
            if (ch != 't')
                distance += 1;
        }
        if (finger == 4) {
            if (ch != 'n')
                distance += 1;
        }
        if (finger == 5) {
            if (ch != 'e') {
                distance += 1;
            }
        }
        if (finger == 6) {
            if (ch != 'i')
                distance += 1;
        }
        return distance;
    }

    // Finding the total distance required by the fingers to press each key of the input
    static void wordDistance(String word, String layout) {
        int distance = 0;
        for (int i = 0; i < word.length(); i++) {
            char ch = word.charAt(i);
            if (layout.equals("QWERTY"))
                distance += findDistanceQWERTY(findFingerQWERTY(ch), ch);
            if (layout.equals("DVORAK"))
                distance += findDistanceDVORAK(findFingerDVORAK(ch), ch);
            if (layout.equals("COLEMAK"))
                distance += findDistanceCOLEMAK(findFingerCOLEMAK(ch), ch);
        }
        System.out.println(distance);
    }

    // Finding the distance moved by each finger separately to press each key and then the average of each
    static void wordFingerDistance(String word, String layout) {
        int[] distance = {0, 0, 0, 0, 0, 0, 0, 0, 0};
        for (int i = 0; i < word.length(); i++) {
            char ch = word.charAt(i);
            if (layout.equals("QWERTY"))
                distance[findFingerQWERTY(ch)] += findDistanceQWERTY(findFingerQWERTY(ch), ch);
            if (layout.equals("DVORAK"))
                distance[findFingerDVORAK(ch)] += findDistanceDVORAK(findFingerDVORAK(ch), ch);
```

```
            if (layout.equals("COLEMAK"))
                distance[findFingerCOLEMAK(ch)] += findDistanceCOLEMAK(findFingerCOLEMAK(ch), ch);
        }
        System.out.println("Layout: " + layout);
        int avDistance = 0;
        for (int i = 0; i < distance.length; i++) {
            System.out.println("Finger " + (i + 1) + ": " + distance[i]);
            avDistance += distance[i];
        }
        // Sum and Average of all fingers
        System.out.println("Total distance: " + avDistance);
        System.out.println("Average of all fingers: " + (avDistance / distance.length));
    }
}
```

Enter sentence:
the at there some my of be use her than and this an would first a have each make water to from which like been in or she him call is one do into who you had how time oil that by their has its it word if look now he but will two find was not up more long for what other write down on all about go day are were out see did as we many number get with when then no come his your them way made they can these could may I said so people part
Distance moved by fingers to type above sentence:

Qwerty:
Layout: QWERTY
Finger 1: 0
Finger 2: 18
Finger 3: 46
Finger 4: 57
Finger 5: 78
Finger 6: 21
Finger 7: 34
Finger 8: 4
Finger 9: 0
Total distance: 258
Average of all fingers: 28

Dvorak:
133
Layout: DVORAK
Finger 1: 0
Finger 2: 0
Finger 3: 0
Finger 4: 37
Finger 5: 41
Finger 6: 24
Finger 7: 18
Finger 8: 13
Finger 9: 0
Total distance: 133
Average of all fingers: 14

Colemak:
159
Layout: COLEMAK
Finger 1: 0
Finger 2: 18

Finger 3: 12
Finger 4: 27
Finger 5: 83
Finger 6: 10
Finger 7: 9
Finger 8: 0
Finger 9: 0
Total distance: 159
Average of all fingers: 17

luckyNumbers.java

```java
// In number theory, a lucky number is a natural number in a set which is generated by a certain "sieve".
// This sieve is similar to the Sieve of Eratosthenes that generates the primes, but it eliminates numbers based on their position in the remaining set,
// instead of their value (or position in the initial set of natural numbers).
// Starting from 2, start eliminating numbers till the count is big enough that all remaining numbers stay

import java.util.Scanner;

public class luckyNumbers {
    int N;
    int[] arr;

    public static void main(String[] args) {
        luckyNumbers obj = new luckyNumbers();
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter value of N");
        obj.N = sc.nextInt();
        sc.close();
        obj.arr = new int[obj.N];
        for (int i = 0; i < obj.N; i++) {
            obj.arr[i] = i + 1;
        }
        obj.print();
        int steps = obj.numberOfSteps();
        for (int i = 1; i <= steps; i++) {
            obj.eliminate(++i);
        }
        System.out.println("Lucky numbers: ");
        obj.print();
    }

    void eliminate(int count) {
        int a = 0;
        for (int i = 0; i < N; i++) {
            if (arr[i] == 0) continue;
            if ((a + 1) % count == 0) arr[a] = 0;
            a += 1;
        }
    }

    int numberOfSteps() {
        // Find number of steps of elimination required to find lucky numbers for a given value of N
        int N = this.N;
        int steps = 0;
        int i = 2;
        while (i <= N) {
            int R = (int) (Math.floor(N / (i + 0.0)));
            N -= R;
            i += 1;
            steps += 1;
        }
        return steps;
    }

    void print() {
        for (int i = 0; i < N; i++) {
            if (arr[i] != 0) System.out.print(arr[i] + " ");
        }
        System.out.println();
    }
}
```

```
Enter value of N
10
1 2 3 4 5 6 7 8 9 10
Lucky numbers:
1 3 5 7 9
Enter value of N
50
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
Lucky numbers:
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
Enter value of N
100
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65
66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96
97 98 99 100
Lucky numbers:
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65
67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

potential.java

```java
/* Finding potential of each word in a sentence, printing words in ascend
ing order of their potential
 * Potential is the sum of positions of each letter in a word */

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class potential {
    public static void main(String[] args) throws IOException {
        // Getting Input
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter sentence in upper case: ");
        String sentence = br.readLine();
        // Variables
        sentence = sentence.trim();
        sentence += " ";
        char ch;
        String word = "";
        int length = sentence.length();
        int sum = 0;
        int counter = 1;
        int[] arrSum = new int[length];
        String[] arrStr = new String[length];
        //Printing potential of each word. Each word and it's sum stored an array
        System.out.println("\nPotential: ");
        for (int i = 0; i < length; i++) {
            ch = sentence.charAt(i);
            if (ch != ' ') {
                sum += ch - 64;
                word += ch;
            } else {
                arrSum[counter] = sum;
                arrStr[counter] = word;
                System.out.println(word + ": " + sum);
                sum = 0;
                word = "";
                counter += 1;
            }
        }
        // Sorting words based on potential using Bubble Sort
        int temp;
        String tem;
        for (int i = 0; i < counter; i++) {
            for (int j = 0; j < (counter - i - 1); j++) {
                if (arrSum[j] > arrSum[j + 1]) {
                    //Swapping potential values
                    temp = arrSum[j];
                    arrSum[j] = arrSum[j + 1];
                    arrSum[j + 1] = temp;
                    //Swapping corresponding word
                    tem = arrStr[j];
                    arrStr[j] = arrStr[j + 1];
                    arrStr[j + 1] = tem;
                }
            }
        }
        //Printing the final values
        System.out.println();
        for (int i = 1; i < counter; i++)
            System.out.print(arrStr[i] + " ");
        System.out.println();
    }
}
/**
 * Variable data table
 * Variable Data type Function
 * sentence String Used to store sentence entered
 * ch char Used in calculations
 * word String Used to store words from sentence
 * length int Used to store length
 * sum int Used in calculations
 * counter int Used as a counter
 * arrSum[] int Used to store potential of each word
 * arrStr[] String Used to store each word
 * temp int Used in swapping in sorting
 */
```

Enter sentence in upper case:
feel your eyes they all over me

Potential:
feel: 156
your: 207
eyes: 182
they: 186
all: 121
over: 188
me: 82

me all feel eyes they over your
Enter sentence in upper case:
dont be shy take control of me

Potential:
dont: 181
be: 71
shy: 148
take: 165
control: 321
of: 85
me: 82

be me of shy take dont control
Enter sentence in upper case:
get the vibe its gonna be a lit tonight

Potential:
get: 128
the: 129
vibe: 166
its: 144
gonna: 211
be: 71
a: 33
lit: 137
tonight: 317

a be get the lit its vibe gonna tonight

recursiveSortSearch.java

```java
import java.util.Scanner;

public class recursiveSortSearch {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter size of array");
        int N = sc.nextInt();
        int[] arr = new int[N];
        System.out.println("Enter " + N + " numbers");
        for (int i = 0; i < N; i++) {
            arr[i] = sc.nextInt();
        }
        System.out.println("Original Array: ");
        print(arr);
        System.out.println("\nSorting using bubble sort");
        bubbleSort(arr, 0, 0);
        print(arr);
        System.out.println("\nEnter element to search for binary search");
        int e = sc.nextInt();
        sc.close();
        int search = binarySearch(arr, 0, N - 1, e);
        if (search == -1) {
            System.out.println("Element not found");
        } else {
            System.out.println("Element found at " + (search+1) + "th index");
        }
    }

    // Bubble Sort
    static void bubbleSort(int[] arr, int i, int j) {
        if (j == arr.length - i - 1) {
            if (i != arr.length - 2) {
                bubbleSort(arr, ++i, 0);
            } else return;
        } else {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
            bubbleSort(arr, i, ++j);
        }
    }

    // Binary Search
    static int binarySearch(int[] arr, int l, int u, int e) {
        if (u >= l) {
            int m = l + (u - l) / 2;
            if (arr[m] == e)
                return m;
            if (arr[m] > e)
                return binarySearch(arr, l, m - 1, e);
            return binarySearch(arr, m + 1, u, e);
        }
        return -1;
    }

    static void print(int[] arr) {
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i] + " ");
        }
    }
}

```

```
Enter size of array
5
Enter 5 numbers
1
5
4
3
2
Original Array:
1 5 4 3 2
Sorting using bubble sort
1 2 3 4 5
Enter element to search for binary search
4
Element found at 4th index
Enter size of array
6
Enter 6 numbers
8
6
4
2
5
10
Original Array:
8 6 4 2 5 10
Sorting using bubble sort
2 4 5 6 8 10
Enter element to search for binary search
4
Element found at 2th index
Enter size of array
3
Enter 3 numbers
6
5
4
Original Array:
6 5 4
Sorting using bubble sort
4 5 6
Enter element to search for binary search
```

7
Element not found

reverseEncryption.java

```java
    // Program which takes a string of a paragraph from user
// The characters of odd sentences are added with number
// The even sentences are reversed in order

import java.util.Scanner;
import java.util.StringTokenizer;

public class reverseEncryption {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter sentence: ");
        String paragraph = sc.nextLine();
        sc.close();
        StringTokenizer st = new StringTokenizer(paragraph, ".");
        String[] sentences = new String[st.countTokens()];
        int i = 0;
        while (st.hasMoreTokens()) {
            sentences[i++] = st.nextToken();
        }
        for (i = 0; i < sentences.length; i++) {
            if (i + 1 % 2 == 0) {
                for (int j = 0; j < sentences[i].length() - 1; j++) {
                    sentences[i] = sentences[i].substring(0, j) +
                            (char)(sentences[i].charAt(j) + 1) + sentences[i].substring(j + 1);
                }
            } else {
                sentences[i] = (new StringBuffer(sentences[i]).reverse()).toString();
            }
        }
        for (i = 0; i < sentences.length; i++) {
            System.out.println(sentences[i]);
        }
    }
}
```

Enter sentence:
Sentence one. Sentence two. Sentence three. Sentence four. Sentence five.
eno ecnetneS
owt ecnetneS
eerht ecnetneS
ruof ecnetneS
evif ecnetneS
Enter sentence:
Hello there. General Kenobi. Nice to meet you. I will destroy you with my lightsabers. Oh we will see about that.
ereht olleH
iboneK lareneG
uoy teem ot eciN
srebasthgil ym htiw uoy yortsed lliw I
taht tuoba ees lliw ew hO

reverseLinkedList.java

```java
// Given pointer to the head node of a linked list,
// the task is to reverse the linked list. We need to
// reverse the list by changing the links between nodes.
class reverseLinkedList {

    static Node head;

    public static void main(String[] args) {
        reverseLinkedList list = new reverseLinkedList();
        head = new Node(85);
        head.next = new Node(15);
        head.next.next = new Node(4);
        head.next.next.next = new Node(20);

        System.out.println("Given Linked list");
        list.printList(head);
        head = list.reverse(head);
        System.out.println();
        System.out.println("Reversed linked list ");
        list.printList(head);
    }

    Node reverse(Node node) {
        Node prev = null;
        Node current = node;
        Node next = null;
        while (current != null) {
            next = current.next;
            current.next = prev;
            prev = current;
            current = next;
        }
        node = prev;
        return node;
    }

    void printList(Node node) {
        while (node != null) {
            System.out.print(node.data + " ");
            node = node.next;
        }
    }

    static class Node {

        int data;
        Node next;

        Node(int d) {
            data = d;
            next = null;
        }
    }
}
```

Given Linked list
85 15 4 20
Reversed linked list
20 4 15 85

sentencePalindromeGenerator.java

```java
// Write a program to accept a sentence which maybe terminated by either '?', '!' or '.' only.
// The words ware to be seperated with single blank space and are in uppercase
// The following tasks are to be performed
// a) Check the validity for the accepted sentence
// b) Convert any non palindrome words into palindromes by concatenating the word by its reverse (excluding the last
// character)
// c) Display the original sentence along with the converted sentence

import java.util.Scanner;

public class sentencePalindromeGenerator {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Taking inputs and task a
        System.out.println("Enter sentence: ");
        String sentence = sc.nextLine();
        String sentence2 = sentence;
        char lastChar = sentence.charAt(sentence.length() - 1);
        if (lastChar != '!' && lastChar != '?' && lastChar != '.') {
            System.out.println("Invalid input! Sentence must terminate with either !,? or .");
            System.exit(0);
        }
        sc.close();
        // Task b
        sentence = sentence.substring(0, sentence.length() - 1).concat(" ").toUpperCase();
        String[] words = sentence.split(" ");
        String newSentence = "";
        for (int i = 0; i < words.length; i++) {
            boolean b = isPalindrome(words[i]);
            if (b == true)
                newSentence += words[i] + " ";
            if (b == false) {
                // Converting non-palindrome to a palindrome word
                String word2 = words[i].substring(0, words[i].length() - 1);
                // Checking if the last character is repeated and fixing it
                if (word2.charAt(word2.length() - 1) == words[i].charAt(words[i].length() - 1))
                    word2 = word2.substring(0, word2.length() - 1);
                // Concatenating the reversed
                StringBuffer sb = new StringBuffer(word2);
                sb.reverse();
                String str2 = sb.toString();
                newSentence += words[i] + str2 + " ";
            }
        }
        // Task c
        System.out.println("\nFinal result: ");
        System.out.println(sentence2 + "\n" + newSentence.trim() + lastChar);
    }

    public static boolean isPalindrome(String word) {
        StringBuffer sb = new StringBuffer(word);
        sb.reverse();
        String str2 = sb.toString();
        return str2.equals(word);
    }
}
```

Enter sentence:
Hello there Gerenal Kenobi!

Final result:
Hello there Gerenal Kenobi!
HELLOLLEH THEREREHT GERENALANEREG KENOBIBONEK!
Enter sentence:
Welcome to my tower Master Chief.

Final result:
Welcome to my tower Master Chief.
WELCOMEMOCLEW TOT MYM TOWEREWOT MASTERETSAM CHIEFEIHC.
Enter sentence:
The Doom Slayer has entered the facility!

Final result:
The Doom Slayer has entered the facility!
THEHT DOOMOOD SLAYEREYALS HASAH ENTEREDERETNE THEHT FACILITYTILICAF!

ticTacToe.java

```java
// TicTacToe game in Java for two players.

import java.util.Scanner;

public class ticTacToe {
    public static Scanner sc = new Scanner(System.in);
    static char[][] board = {
            {'1', '2', '3'},
            {'4', '5', '6'},
            {'7', '8', '9'},
    };
    static char player1choice, player2choice;
    static int row, column;
    static int player1count = 0, player2count = 0;

    public static void main(String[] args) {
        System.out.println("Welcome to the tictactoe game made by Kavin!");
        System.out.println("Enter letter to represent player 1: ");
        player1choice = sc.next().charAt(0);
        System.out.println("Enter letter to represent player 2: ");
        player2choice = sc.next().charAt(0);
        printBoard();
        // If no player has won, continue
        while (!checkWinner(player1choice) && !checkWinner(player2choice)) {
            player1Turn();
            player2Turn();
        }
    }

    // Print the current board
    public static void printBoard() {
        for (int i = 0; i < 3; i++) {
            System.out.print("| ");
            for (int j = 0; j < 3; j++) {
                System.out.print(board[i][j] + " | ");
            }
            System.out.println();
        }
    }

    // Determine 2D (x,y) coordinates from 1D position
    public static void determineRowAndColumn(int position) {
        if (position == 1) {
            row = 0;
            column = 0;
        }
        if (position == 2) {
            row = 0;
            column = 1;
        }
        if (position == 3) {
            row = 0;
            column = 2;
        }
        if (position == 4) {
            row = 1;
            column = 0;
        }
        if (position == 5) {
            row = 1;
            column = 1;
        }
        if (position == 6) {
            row = 1;
            column = 2;
        }
        if (position == 7) {
            row = 2;
            column = 0;
        }
        if (position == 8) {
            row = 2;
            column = 1;
        }
        if (position == 9) {
            row = 2;
            column = 2;
        }
    }

```

```java
    // Make the move on the board for player 1
    public static void player1move(int row, int column) {
        board[row][column] = player1choice;
    }

    // Make the move on the board for player 2
    public static void player2move(int row, int column) {
        board[row][column] = player2choice;
    }

    // Check if a player won
    public static boolean checkWinner(char playerChoice) {
        // Check rows
        if (board[0][0] == playerChoice && board[0][1] == playerChoice && board[0][2] == playerChoice)
            return true;
        if (board[1][0] == playerChoice && board[1][1] == playerChoice && board[1][2] == playerChoice)
            return true;
        if (board[2][0] == playerChoice && board[2][1] == playerChoice && board[2][2] == playerChoice)
            return true;
        // Check columns
        if (board[0][0] == playerChoice && board[1][0] == playerChoice && board[2][0] == playerChoice)
            return true;
        if (board[0][1] == playerChoice && board[1][1] == playerChoice && board[2][1] == playerChoice)
            return true;
        if (board[0][2] == playerChoice && board[1][2] == playerChoice && board[2][2] == playerChoice)
            return true;
        // Check diagonals
        if (board[0][0] == playerChoice && board[1][1] == playerChoice && board[2][2] == playerChoice)
            return true;
        return board[0][2] == playerChoice && board[1][1] == playerChoice && board[2][0] == playerChoice;
    }

    public static void checkTie() {
        int count = player1count + player2count; // Number of turns taken by players
        if (count == 9) {
            System.out.println("It's a tie");
            System.exit(0);
        }
    }

    public static void player1Turn() {
        int flag1 = 0;
        while (flag1 == 0) {
            System.out.println("Player " + player1choice + ", enter position {1..9}");
            int player1position = sc.nextInt();
            determineRowAndColumn(player1position);
            if (board[row][column] != player2choice && board[row][column] != player1choice) {
                player1count += 1;
                checkTie();
                player1move(row, column);
                printBoard();
                boolean b = checkWinner(player1choice);
                if (b) {
                    System.out.println("Player 1(" + player1choice + ") won!");
                    System.exit(0);
                }
                flag1 = 1;
            } else {
                System.out.println("Already occupied! Try again");
            }
        }
    }

    public static void player2Turn() {
        int flag2 = 0;
        while (flag2 == 0) {
            System.out.println("Player " + player2choice + ", enter position {1..9}");
            int player2position = sc.nextInt();
            determineRowAndColumn(player2position);
            if (board[row][column] != player1choice && board[row][column] != player2choice) {
                player2count += 1;
                checkTie();
                player2move(row, column);
                printBoard();
                boolean b = checkWinner(player2choice);
                if (b) {
                    System.out.println("Player 2 (" + player2choice + ") won!");
                    System.exit(0);
                }
                flag2 = 1;
            } else {
                System.out.println("Already occupied! Try again");
            }
```

```
        }
    }
}
/**
 * Variable            Data        Table
 * board               int[][]     Store board
 * player1choice       char        Store 1st users avatar
 * player2choice       char        Store 2nd users avatar
 * player1position     int         Where 1st user choose to move
 * player2position     int         Where 2nd user choose to move
 * row, column         int         2d coordinates for move
 * position            int         1d coordinates for move
 * i, j                int         Used in calculation
 */
```

```
Welcome to the tictactoe game made by Kavin!
Enter letter to represent player 1:
X
Enter letter to represent player 2:
O
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
Player X, enter position {1..9}
1
| X | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
Player O, enter position {1..9}
5
| X | 2 | 3 |
| 4 | O | 6 |
| 7 | 8 | 9 |
Player X, enter position {1..9}
3
| X | 2 | X |
| 4 | O | 6 |
| 7 | 8 | 9 |
Player O, enter position {1..9}
6
| X | 2 | X |
| 4 | O | O |
| 7 | 8 | 9 |
Player X, enter position {1..9}
2
| X | X | X |
| 4 | O | O |
| 7 | 8 | 9 |
Player 1(X) won!
```

towerOfHanoi.java

```java
/*
Iterative Solution of Tower of Hanoi using Stack
The Tower of Hanoi is a mathematical puzzle. It consists of three poles
and a number of disks of different sizes which can slide onto any poles.
The puzzle starts with the disk in a neat stack in ascending order of size
in one pole, the smallest at the top thus making a conical shape. The objective
of the puzzle is to move all the disks from one pole (say 'source pole') to another
pole (say 'destination pole') with the help of the third pole (say auxiliary pole).
The puzzle has the following two rules:
      1. You can't place a larger disk onto a smaller disk
      2. Only one disk can be moved at a time
 */
public class towerOfHanoi {
    public static void main(String[] args) {
        int disks = 3;
        towerOfHanoi toh = new towerOfHanoi();
        // Create three stacks of size disk to hold the disks
        Stack src = toh.create(disks);
        Stack dest = toh.create(disks);
        Stack aux = toh.create(disks);
        toh.solution(disks, src, aux, dest);
    }

    Stack create(int size) {
        Stack stack = new Stack();
        stack.size = size;
        stack.top = -1;
        stack.arr = new int[size];
        return stack;
    }

    boolean isFull(Stack stack) {
        return (stack.top == stack.size - 1);
    }

    boolean isEmpty(Stack stack) {
        return (stack.top == -1);
    }

    // Adding an item to the stack.
    void push(Stack stack, int n) {
        if (isFull(stack)) return;
        stack.arr[++stack.top] = n;
    }

    // Removing an item from the top
    int pop(Stack stack) {
        if (isEmpty(stack))
            return Integer.MIN_VALUE;
        return stack.arr[stack.top--];
    }

    void moveDisksBetweenPoles(Stack src, Stack dest, char s, char d) {
        int pole1TopDisk = pop(src);
        int pole2TopDisk = pop(dest);

        // When pole 1 is empty
        if (pole1TopDisk == Integer.MIN_VALUE) {
            push(src, pole2TopDisk);
            moveDisk(d, s, pole2TopDisk);
        }

        // When pole2 pole is empty
        else if (pole2TopDisk == Integer.MIN_VALUE) {
            push(dest, pole1TopDisk);
            moveDisk(s, d, pole1TopDisk);
        }

        // When top disk of pole1 > top disk of pole2
        else if (pole1TopDisk > pole2TopDisk) {
            push(src, pole1TopDisk);
            push(src, pole2TopDisk);
            moveDisk(d, s, pole2TopDisk);
        }
        // When top disk of pole1 < top disk of pole2
        else {
            push(dest, pole2TopDisk);
            push(dest, pole1TopDisk);
            moveDisk(s, d, pole1TopDisk);
        }
```

```java
    }

    void moveDisk(char fromRod, char toRod, int disk) {
        System.out.println("Move disk " + disk + " from " + fromRod + " to " + toRod);
    }

    void solution(int disks, Stack src, Stack aux, Stack dest) {
        int i, noOfMoves;
        char s = 'S', d = 'D', a = 'A';
        // If number of disks is even, then interchange destination pole and auxiliary pole
        if (disks % 2 == 0) {
            char tmp = d;
            d = a;
            a = tmp;
        }
        noOfMoves = (int) (Math.pow(2, disks) - 1);

        // Larger disks will be pushed first
        for (i = disks; i >= 1; i--) {
            push(src, i);
        }

        for (i = 1; i <= noOfMoves; i++) {
            if (i % 3 == 1) moveDisksBetweenPoles(src, dest, s, d);
            else if (i % 3 == 2) moveDisksBetweenPoles(src, aux, s, a);
            else if (i % 3 == 0) moveDisksBetweenPoles(aux, dest, a, d);
        }
    }

    class Stack {
        int size;
        int top;
        int[] arr;
    }
}
```

Move disk 1 from S to D
Move disk 2 from S to A
Move disk 1 from D to A
Move disk 3 from S to D
Move disk 1 from A to S
Move disk 2 from A to D
Move disk 1 from S to D

typingTest.java

```java
// Program of a typing test to find speed in WPM, CPM, accuracy % based on commonly typed words in English 1k.

import java.time.LocalTime;
import java.util.Scanner;

public class typingTest {
    static String input;
    static int WPM, CPM, numberOfWords, correctWords;
    static double start, end, time, accuracy, minutesTaken;
    // Class Variables
    String[] bank = {"be", "have", "do", "say", "get", "make", "go", "know", "take",
            "see", "come", "think", "look", "want", "give", "use", "find", "tell",
            "ask", "work", "seem", "feel", "try", "leave", "call", "person", "life",
            "day", "number", "again"};

    typingTest() {
        input = "";
        WPM = CPM = numberOfWords = correctWords = 0;
        start = end = time = accuracy = minutesTaken = 0.0;
    }

    public static void main(String[] args) {
        typingTest obj = new typingTest();
        obj.countdown();
        obj.input();
        obj.calculate();
        obj.display();
    }

    void countdown() {
        try {
            System.out.println("3!");
            Thread.sleep(1000);
            System.out.println("2!");
            Thread.sleep(1000);
            System.out.println("1!");
            Thread.sleep(1000);
            System.out.println("Start!\n");
            // Test words
            System.out.println("be have do say get make go know take see");
            System.out.println("come think look want give use fine tell ask work");
            System.out.println("seem feel try leave call person life day number again\n");
        } catch (InterruptedException e) {
            System.out.println("Error!");
        }
    }

    void input() {
        Scanner sc = new Scanner(System.in);
        start = LocalTime.now().toNanoOfDay();
        input = sc.nextLine();
        end = LocalTime.now().toNanoOfDay();
        time = (end - start) / 1000000000.0; // Convert nanoseconds to seconds
        sc.close();
    }

    void calculate() {
        numberOfWords = 30;
        Scanner words = new Scanner(input);
        int i = 0;
        while (words.hasNext()) {
            if (bank[i].equals(words.next()))
                ++correctWords;
            ++i;
        }
        words.close();
        minutesTaken = time / 60.0;
        WPM = (int) ((correctWords * 1.0) / minutesTaken);
        accuracy = ((correctWords * 1.0) / numberOfWords) * 100.0;
        accuracy = Math.round(accuracy * 100.0) / 100.0;
        CPM = (int) ((114.0 * (accuracy / 100.0)) / minutesTaken);
    }

    void display() {
        System.out.println("WPM: " + WPM);
        System.out.println("CPM: " + CPM);
        System.out.println("Accuracy % : " + accuracy);
    }
}
/**
```

```
81    * Variable        Data        Table
82    * input           String      Sentence entered by user
83    * bank            String[]    Word bank to be entered
84    * WPM, CPM        int         (words per minute and characters per minute) Store speed of user
85    * numberOfWords   int         Store Total number of words
86    * correctWords    int         Number of words correct by user
87    * start, end      int         Store nano seconds of day
88    * time            int         end - start (Store time taken)
89    * minutesTaken    int         Convert time to minutes
90    * accuracy        int         Accuracy in percentage
91    */
```

3!
2!
1!
Start!

be have do say get make go know take see
come think look want give use fine tell ask work
seem feel try leave call person life day number again

be have do say get make go know take see come think look want give use fine tell ask work seem feel try leave call person life day number again
WPM: 112
CPM: 428
Accuracy % : 96.67

verticalBanners.java

```java
/* The names of the teams participating in a competition should be
displayed on a banner vertically, to accommodate as many teams as
possible in a single banner. Design a program to accept the names
of N teams, where 2 < N < 9 and display them in vertical order,
side by side with a horizontal tab (i.e. eight spaces). */

import java.util.Scanner;

public class verticalBanners {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("ENTER THE VALUE OF N: ");
        int n = in.nextInt();
        in.nextLine();

        if (n <= 2 || n >= 9) {
            System.out.println("INVALID INPUT");
            return;
        }

        String[] teams = new String[n];
        int highLen = 0;

        for (int i = 0; i < n; i++) {
            System.out.print("Team " + (i + 1) + ": ");
            teams[i] = in.nextLine();
            if (teams[i].length() > highLen) {
                highLen = teams[i].length();
            }
        }

        for (int i = 0; i < highLen; i++) {
            for (int j = 0; j < n; j++) {
                int len = teams[j].length();
                if (i >= len) {
                    System.out.print(" \t");
                } else {
                    System.out.print(teams[j].charAt(i) + "\t");
                }
            }
            System.out.println();
        }
    }
}
```

ENTER THE VALUE OF N: 3
Team 1: BRUH
Team 2: LMFAO
Team 3: ROFLMAO

B	L	R
R	M	O
U	F	F
H	A	L
	O	M
		A
		O

ENTER THE VALUE OF N: 2
INVALID INPUT
ENTER THE VALUE OF N: 4
Team 1: DUA LIPA
Team 2: TAYLOR SWIFT
Team 3: DOJA CAT
Team 4: HALSEY

D	T	D	H
U	A	O	A
A	Y	J	L
	L	A	S
L	O		E
I	R	C	Y
P		A	
A	S	T	
	W		
	I		
	F		
	T		

wordShift.java

```java
import java.util.Scanner;
import java.util.StringTokenizer;

public class wordShift {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter sentence: ");
        String sentence = sc.nextLine();
        StringTokenizer st = new StringTokenizer(sentence); // Default delimiter is used ' '
        System.out.println("Enter shift: ");
        int shift = sc.nextInt() % st.countTokens(), count = 0;
        sc.close();
        String s2 = "", s3 = "";
        // Append words to strings
        while (st.hasMoreTokens()) {
            if (count < st.countTokens() - shift) {
                s2 = s2 + st.nextToken() + " ";
                count++;
            } else
                s3 += st.nextToken() + " ";
        }
        System.out.println("Shifted sentence is:\n" + s3 + s2);
    }
}
```

Enter sentence:
achilles was a great warrior
Enter shift:
3
Shifted sentence is:
was a great warrior achilles
Enter sentence:
a powerful body requires a powerful mind
Enter shift:
2
Shifted sentence is:
requires a powerful mind a powerful body
Enter sentence:
maximum efficiency with minimum effort is my style
Enter shift:
4
Shifted sentence is:
with minimum effort is my style maximum efficiency

wordleStartChecker.java

```java
// Deterministic factor for wordle starter accuracy:
// Does it include all most commonly used characters for its size?

import java.util.Scanner;

public class wordleStartChecker {
    static char[] list = {'e', 'a', 'r', 'i', 'o', 't', 'n', 's', 'l', 'c',
            'u', 'd', 'p', 'm', 'h', 'g', 'b', 'f', 'y', 'w', 'k', 'v', 'x', 'z', 'j', 'q'};
    static double[] freq = {56.88, 43.31, 38.64, 38.45, 36.51, 35.43, 33.92, 29.23,
            27.98, 23.13, 18.51, 17.25, 16.14, 15.36, 15.31, 12.59, 10.56, 9.24, 9.06,
            6.57, 5.61, 5.13, 1.48, 1.39, 1, 1};
    static int ntries;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("How many test 5-letter words do you want?");
        ntries = sc.nextInt();
        String word = "";
        for (int i = 1; i <= ntries; i++) {
            word += sc.next().trim();
        }
        compareWord(word.toLowerCase());
        sc.close();
    }

    static void compareWord(String word) {
        int length = word.length();
        // Create word with the highest accuracy for given length
        String check = "";
        int checkPercent = 0;
        for (int i = 0; i < length; i++) {
            check += list[i];
            checkPercent += freq[i];
        }
        // Check if word and check contains the same letters
        int flag = 0, wordPercent = 0;
        for (int i = 0; i < length; i++) {
            if (check.contains("" + word.charAt(i))) {
                flag += 1;
            }
            wordPercent += freq[findIndex(word.charAt(i))];
        }
        System.out.println(word + "\n" + check);
        if (flag == length) System.out.println("Your word(s) is an ideal wordle starter");
        if (flag < length) {
            double percent = (wordPercent * 100.0) / checkPercent;
            System.out.println("Your word isn't the ideal wordle starter. It is " +
                    Math.round((percent / ntries) * Math.pow(10, 2)) / Math.pow(10, 2) + "% of an ideal world");
        }
    }

    static int findIndex(char ch) {
        for (int i = 0; i < 26; i++) {
            if (list[i] == ch) return i;
        }
        return 0;
    }
}
```

How many test 5-letter words do you want?
1
audio
audio
eario
Your word isn't the ideal wordle starter. It is 72.04% of an ideal world
How many test 5-letter words do you want?
1
crate
crate
eario
Your word isn't the ideal wordle starter. It is 92.42% of an ideal world
How many test 5-letter words do you want?
3
siren
octal
dumpy
sirenoctaldumpy
eariotnslcudpmh
Your word isn't the ideal wordle starter. It is 32.88% of an ideal world