Details

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

Acknowledgement

I would like to thank my CS Teacher Mrs. Mithila Dargan for supporting me throughout the creation of this project, teaching me the wonderful subject of Computer Science and the importance of hard work. I would also like to thank my friends and family since this wouldn't be possible without them.

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

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.