**MODULE -1**

**LAB EXERCISE PROGRAMS – INSERTION SORT, STRONG PASSWORD**

1. **INSERTION SORT**

// C program for insertion sort

#include <math.h>

#include <stdio.h>

/\* Function to sort an array

using insertion sort\*/

void insertionSort(int arr[], int n)

{

int i, key, j;

for (i = 1; i < n; i++)

{

key = arr[i];

j = i - 1;

/\* Move elements of arr[0..i-1],

that are greater than key,

to one position ahead of

their current position \*/

while (j >= 0 && arr[j] > key)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

// A utility function to print

// an array of size n

void printArray(int arr[], int n)

{

int i;

for (i = 0; i < n; i++)

printf("%d ", arr[i]);

printf("\n");

}

// Driver code

int main()

{

int arr[] = {12, 11, 13, 5, 6};

int n = sizeof(arr) / sizeof(arr[0]);

insertionSort(arr, n);

printArray(arr, n);

return 0;

}

**OUTPUT : 5 6 11 12 13**

1. **STRONG PASSWORD**

#include <stdio.h>

#include <string.h>

// Function to determine the minimum number of characters to add

int minimumNumber(int n, char \*password) {

int required\_chars = 0;

int has\_digit = 0, has\_lower = 0, has\_upper = 0, has\_special = 0;

const char \*special\_characters = "!@#$%^&\*()-+";

// Check the existing characters in the password

for (int i = 0; i < n; i++) {

if (password[i] >= '0' && password[i] <= '9') has\_digit = 1;

else if (password[i] >= 'a' && password[i] <= 'z') has\_lower = 1;

else if (password[i] >= 'A' && password[i] <= 'Z') has\_upper = 1;

else if (strchr(special\_characters, password[i])) has\_special = 1;

}

// Count the missing types of characters

if (!has\_digit) required\_chars++;

if (!has\_lower) required\_chars++;

if (!has\_upper) required\_chars++;

if (!has\_special) required\_chars++;

// Ensure the password length is at least 6 characters

if (n + required\_chars < 6) {

required\_chars += (6 - (n + required\_chars));

}

return required\_chars;

}

int main() {

int n;

char password[101];

// Input the length of the password and the password itself

scanf("%d", &n);

scanf("%s", password);

// Calculate and print the minimum number of characters to add

int result = minimumNumber(n, password);

printf("%d\n", result);

return 0;

}

**INPUT:** 3

Ab1

**OUTPUT:** Number of characters to make password strong: 3

5

aB1@

**OUTPUT:** Number of characters to make password strong: 1

**ADDITIONAL PROGRAMS – THE POWER SUM, RUNNING TIME OF ALGORITHM**

**3. THE POWER SUM**

#include <stdio.h>

#include <math.h>

// Function to recursively find the power sums

int powerSumHelper(int X, int N, int num) {

int power = pow(num, N);

if (power > X) {

return 0;

} else if (power == X) {

return 1;

} else {

return powerSumHelper(X - power, N, num + 1) + powerSumHelper(X, N, num + 1);

}

}

// Main function to find the number of ways to express X as sum of N-th powers of unique natural numbers

int powerSum(int X, int N) {

return powerSumHelper(X, N, 1);

}

int main() {

int X, N;

printf("Enter X: ");// Input the values of X and N

scanf("%d", &X);

printf("Enter N: ");

scanf("%d", &N);

int result = powerSum(X, N); // Calculate and print the number of combinations

printf("%d\n", result);

return 0;

}

**Input:**

Enter X: 100

Enter N: 3

**Output: 1**

**Explanation:**

The only way to express 100 as the sum of cubes of unique natural numbers is: 100=43100 = 4^3100=43

**4.RUNNING TIME OF ALGORITHM**

**#include <stdio.h>**

// Function to perform Insertion Sort and count the number of shifts

int runningTime(int arr[], int n) {

int shifts = 0;

for (int i = 1; i < n; i++) {

int key = arr[i];

int j = i - 1;

// Move elements of arr[0..i-1] that are greater than key

//to one position ahead of their current position

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j = j - 1;

shifts++;

}

arr[j + 1] = key;

}

return shifts;

}

int main() {

int n;

// Read the number of elements

printf("Enter the number of elements: ");

scanf("%d", &n);

int arr[n];

printf("Enter the elements: "); // Read the array elements

for (int i = 0; i < n; i++) {

scanf("%d", &arr[i]);

}

int result = runningTime(arr, n); // Get the number of shifts and print it

printf("Number of shifts: %d\n", result);

return 0;

}

**Input :** Enter the number of elements: 5

Enter the elements: 2 1 3 1 2

**Output :**Number of shifts: 4

**MODULE -2**

**LAB EXERCISE PROGRAMS – Sorting: Comparator , Pattern-syntax-checker**

1. **Sorting Comparator**

import java.util.\*;

class Player {

    String name;

    int score;

    Player(String name, int score) {

        this.name = name;

        this.score = score;

    }

}

class Checker implements Comparator<Player> {

    // complete this method

     public int compare(Player p1, Player p2) {

return p1.score != p2.score ? (p2.score - p1.score) : p1.name.compareTo(p2.name);

    }

}

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        Player[] player = new Player[n];

        Checker checker = new Checker();

        for(int i = 0; i < n; i++){

            player[i] = new Player(scan.next(), scan.nextInt());

        }

        scan.close();

        Arrays.sort(player, checker);

        for(int i = 0; i < player.length; i++){

            System.out.printf("%s %s\n", player[i].name, player[i].score);

        }

    }

}

**Sample Input**

5

amy 100

david 100

heraldo 50

aakansha 75

aleksa 150

**Sample Output**

aleksa 150

amy 100

david 100

aakansha 75

heraldo 50

1. **Pattern-syntax-checker**

import java.util.Scanner;

import java.util.regex.Pattern;

import java.util.regex.PatternSyntaxException;

public class PatternSyntaxChecker {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number of patterns to check:");

int testCases = Integer.parseInt(scanner.nextLine());

while (testCases > 0) {

String pattern = scanner.nextLine();

try {

Pattern.compile(pattern);

System.out.println("Valid");

} catch (PatternSyntaxException e) {

System.out.println("Invalid");

}

testCases--;

}

scanner.close();

}

}

**Sample Input**

3

([A-Z])(.+)

[AZ[a-z](a-z)

batcatpat(nat

**Sample Output**

Valid

Invalid

Invalid

**Additional Programs – JAVA SHA-256 , Java Regex 2 - Duplicate Words**

**3.Java SHA-256**

import java.io.\*;

import java.util.\*;

import java.security.\*;

public class Solution {

    public static void main(String[] args) {

        /\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

        Scanner scanner = new Scanner(System.in);

        String key = scanner.next();

        try{

        MessageDigest md = MessageDigest.getInstance("SHA-256");

            md.update(key.getBytes());

                byte[] digest = md.digest();

                StringBuffer stringbuffer = new StringBuffer();

        for (byte b: digest)

        {  // needed to print it in hexadecimal format

            stringbuffer.append(String.format("%02x", b));

        }

                System.out.println(stringbuffer.toString());

                }

         catch (NoSuchAlgorithmException exception)

        {

            System.out.println(exception);

        }

    }

}

**Sample Input**

HelloWorld

**Sample Output**

872e4e50ce9990d8b041330c47c9ddd11bec6b503ae9386a99da8584e9bb12c4

**4. Java Regex 2 - Duplicate Words**

import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class DuplicateWords {

    public static void main(String[] args) {

        String regex = "\\b(\\w+)(?:\\W+\\1\\b)+";

        Pattern p = Pattern.compile(regex, Pattern.CASE\_INSENSITIVE);

         Scanner in = new Scanner(System.in);

        int numSentences = Integer.parseInt(in.nextLine());

          while (numSentences-- > 0) {

            String input = in.nextLine();

                        Matcher m = p.matcher(input);

             // Check for subsequences of input that match the compiled pattern

            while (m.find()) {

                 input = input.replaceAll(m.group(), m.group(1));

            }

            // Prints the modified sentence.

            System.out.println(input);

        }

                in.close();

    }

}

**Sample Input**

5

Goodbye bye bye world world world

Sam went went to to to his business

Reya is is the the best player in eye eye game

in inthe

Hello hello Ab aB

**Sample Output**

Goodbye bye world

Sam went to his business

Reya is the best player in eye game

in inthe

Hello Ab