**A Micro Project Report**

**on**

**Problem Solving using C Language**

Submitted by

#### K.Sushmaswaraj (24475A0545)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**NARASARAOPETA ENGINEERING COLLEGE: NARASARAOPET (AUTONOMOUS)**

**Accredited by NAAC with A+ Grade and NBA under Tier-1**

**NIRF rank in the band of 201-300 and is an ISO 9001:2015 certified Approved by AICTE, New Delhi, Permanently affiliated to JNTU Kakinada, Approved by AICTE, Accredited by NBA and accredited ’A+’ grade by NAAC Narasaraopet-522601, Palnadu(Dt.), Andhra Pradesh, India**

**2024-20****25**

**NARASARAOPETA ENGINEERING COLLEGE: NARASARAOPET**

**(AUTONOMOUS)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

**This is to certify that** K.Sushmaswaraj **rollno : 24475A0545, a Second Year Student of the Department of Computer Science and Engineering, has completed the Micro Project Satisfactorily in “Problem Solving using C Language" for the Academic Year 2024-2025.**.

Project Co-Ordinator HEAD OF THE DEPARTMENT

**Dr. Rama Krishna. Eluri, M.Tech., Ph.D.** **Dr. S. N. Tirumala Rao,** **M.Tech., Ph.D. Asst. Professor Professor**

|  |  |
| --- | --- |
| **S.No** | **Description** |
|  | Read Records of n Different Students in Structure & Sort on the Basis of Marksin Ascending Order |
|  | Employee Record in Descending Order by Age in Structure |
|  | C Program to Convert Roman Number to Decimal Number |
| 4 | Write a program for a matchstick game being played between the computer and a user. Your program should ensure that the computer always wins. Rules  for the game are as follows:  There are 21 matchsticks.  The computer asks the player to pick 1, 2, 3, or 4 matchsticks.  After the person picks, the computer does its picking.  Whoever is forced to pick up the last matchstick loses the game. |

**C programming code**

**Aim 1**

Read Records of n Different Students in Structure & Sort on the Basis of Marksin Ascending Order

**Code**

#include <stdio.h>

#include <string.h>

// Define a structure to hold student information

struct Student {

char name[50];

int roll\_no;

float marks;

}

// Function to sort students based on marks in ascending order

void sortStudents(struct Student students[], int n) {

struct Student temp;

for (int i = 0; i < n - 1; i++) {

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

if (students[i].marks > students[j].marks) {

// Swap the students

temp = students[i];

students[i] = students[j];

students[j] = temp;

}

}

}

}

int main() {

int n;

// Read number of students

printf("Enter number of students: ");

scanf("%d", &n);

// Declare an array of structures to store student data

struct Student students[n];

// Read student details

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

printf("\nEnter details for student %d\n", i + 1);

printf("Enter name: ");

getchar(); // To consume the leftover newline character from previous input

fgets(students[i].name, sizeof(students[i].name), stdin);

students[i].name[strcspn(students[i].name, "\n")] = 0; // Remove the newline character

printf("Enter roll number: ");

scanf("%d", &students[i].roll\_no);

printf("Enter marks: ");

scanf("%f", &students[i].marks);

}

// Sort students based on marks

sortStudents(students, n);

// Display sorted student details

printf("\nSorted student details based on marks (ascending order):\n");

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

printf("\nStudent %d\n", i + 1);

printf("Name: %s\n", students[i].name);

printf("Roll Number: %d\n", students[i].roll\_no);

printf("Marks: %.2f\n", students[i].marks);

}

return 0;

}

**Input :**

Enter number of students: 3

Enter details for student 1

Enter name: John Doe

Enter roll number: 1

Enter marks: 85

Enter details for student 2

Enter name: Jane Smith

Enter roll number: 2

Enter marks: 90

Enter details for student 3

Enter name: Bob Johnson

Enter roll number: 3

Enter marks: 78

**Output :**

Sorted student details based on marks (ascending order):

Student 1

Name: Bob Johnson

Roll Number: 3

Marks: 78.00

Student 2

Name: John Doe

Roll Number: 1

Marks: 85.00

Student 3

Name: Jane Smith

Roll Number: 2

Marks: 90.00

**Aim 2:**

Employee Record in Descending Order by Age in Structure

**Code:**

#include <stdio.h>

#include <string.h>

// Define a structure to hold employee information

struct Employee {

char name[50];

int id;

int age;

}

// Function to sort employees based on age in descending order

void sortEmployees(struct Employee employees[], int n) {

struct Employee temp;

for (int i = 0; i < n - 1; i++) {

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

if (employees[i].age < employees[j].age) { // Change this condition to sort in descending order

// Swap the employees

temp = employees[i];

employees[i] = employees[j];

employees[j] = temp;

}

}

}

}

int main() {

int n;

// Read number of employees

printf("Enter number of employees: ");

scanf("%d", &n);

// Declare an array of structures to store employee data

struct Employee employees[n];

// Read employee details

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

printf("\nEnter details for employee %d\n", i + 1);

printf("Enter name: ");

getchar(); // To consume the leftover newline character from previous input

fgets(employees[i].name, sizeof(employees[i].name), stdin);

employees[i].name[strcspn(employees[i].name, "\n")] = 0; // Remove the newline character

printf("Enter ID: ");

scanf("%d", &employees[i].id);

printf("Enter age: ");

scanf("%d", &employees[i].age);

}

// Sort employees based on age in descending order

sortEmployees(employees, n);

// Display sorted employee details

printf("\nSorted employee details (by age, descending order):\n");

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

printf("\nEmployee %d\n", i + 1);

printf("Name: %s\n", employees[i].name);

printf("ID: %d\n", employees[i].id);

printf("Age: %d\n", employees[i].age);

}

return 0;

}

**Input :**

Enter number of employees: 3

Enter details for employee 1

Enter name: John Doe

Enter ID: 101

Enter age: 32

Enter details for employee 2

Enter name: Jane Smith

Enter ID: 102

Enter age: 45

Enter details for employee 3

Enter name: Bob Johnson

Enter ID: 103

Enter age: 28

**Output :**

Sorted employee details (by age, descending order):

Employee 1

Name: Jane Smith

ID: 102

Age: 45

Employee 2

Name: John Doe

ID: 101

Age: 32

Employee 3

Name: Bob Johnson

ID: 103

Age: 28

**Aim 3:**

C Program to Convert Roman Number to Decimal Number

**Code:**

#include <stdio.h>

#include <string.h>

// Function to get the value of a Roman numeral character

int romanToDecimal(char c) {

switch(c) {

case 'I': return 1;

case 'V': return 5;

case 'X': return 10;

case 'L': return 50;

case 'C': return 100;

case 'D': return 500;

case 'M': return 1000;

default: return 0; // Invalid Roman character

}

}

// Function to convert Roman numeral to Decimal number

int convertRomanToDecimal(char roman[]) {

int decimal = 0;

int i;

// Iterate through the Roman numeral string

for (i = 0; i < strlen(roman); i++) {

// Get the value of the current Roman numeral

int currentVal = romanToDecimal(roman[i]);

// If the current numeral is smaller than the next one, subtract it (e.g., IV = 5 - 1)

if (i + 1 < strlen(roman) && currentVal < romanToDecimal(roman[i + 1])) {

decimal -= currentVal;

} else {

decimal += currentVal;

}

}

return decimal;

}

int main() {

char roman[20];

// Input Roman numeral

printf("Enter a Roman numeral: ");

scanf("%s", roman);

// Convert Roman numeral to decimal and print result

int decimal = convertRomanToDecimal(roman);

printf("The decimal value of %s is: %d\n", roman, decimal);

return 0;

}

**Input 1:**

Enter a Roman numeral: III

**Output 1:**

The decimal value of III is: 3

**Input 2:**

Enter a Roman numeral: IV

**Output 2:**

The decimal value of IV is: 4

**Aim 4:**

Write a program for a matchstick game being played between the computer and a user. Your program should ensure that the computer always wins. Rules

for the game are as follows:

there are 21 matchsticks.

The computer asks the player to pick 1, 2, 3, or 4 matchsticks.

After the person picks, the computer does its picking.

Whoever is forced to pick up the last matchstick loses the game.

**Code**:

#include <stdio.h>

int main() {

int matchsticks = 21; // Total number of matchsticks

int player\_pick, computer\_pick;

printf("Welcome to the Matchstick Game!\n");

printf("There are 21 matchsticks.\n");

printf("The player and computer take turns to pick 1, 2, 3, or 4 matchsticks.\n");

printf("Whoever is forced to pick the last matchstick loses the game.\n");

// Game loop

while (matchsticks > 1) {

// Player's turn

printf("\nThere are %d matchsticks left.\n", matchsticks);

printf("How many matchsticks would you like to pick (1-4)? ");

scanf("%d", &player\_pick);

// Validate player input

if (player\_pick < 1 || player\_pick > 4) {

printf("Invalid choice! Please pick a number between 1 and 4.\n");

continue;

}

// Ensure player does not pick more matchsticks than are remaining

if (player\_pick > matchsticks) {

printf("You can't pick more matchsticks than are left. Please try again.\n");

continue;

}

matchsticks -= player\_pick;

// Check if the player lost

if (matchsticks == 1) {

printf("\nThere is 1 matchstick left.\n");

printf("You were forced to pick the last matchstick. You lose!\n");

break;

}

// Computer's turn (computer always wins)

printf("\nThere are %d matchsticks left.\n", matchsticks);

// To always win, the computer calculates its optimal move

// The computer picks such that the remaining matchsticks after its turn

// is a multiple of 5 (if possible).

// This ensures the computer will eventually win.

// Calculate the computer's optimal pick

computer\_pick = (matchsticks - 1) % 5;

if (computer\_pick == 0) {

// If the matchsticks are a multiple of 5, pick 1 to stay in control

computer\_pick = 1;

}

printf("Computer picks %d matchstick(s).\n", computer\_pick);

matchsticks -= computer\_pick;

// Check if the computer lost

if (matchsticks == 1) {

printf("\nThere is 1 matchstick left.\n");

printf("The computer was forced to pick the last matchstick. You win!\n");

break;

}

}

return 0;

}

**Input :**

Welcome to the Matchstick Game!

There are 21 matchsticks.

The player and computer take turns to pick 1, 2, 3, or 4 matchsticks.

Whoever is forced to pick the last matchstick loses the game.

There are 21 matchsticks left.

How many matchsticks would you like to pick (1-4)? 3

There are 18 matchsticks left.

Computer picks 2 matchstick(s).

There are 16 matchsticks left.

How many matchsticks would you like to pick (1-4)? 4

There are 12 matchsticks left.

Computer picks 1 matchstick(s).

There are 11 matchsticks left.

How many matchsticks would you like to pick (1-4)? 2

There are 9 matchsticks left.

Computer picks 4 matchstick(s).

There are 5 matchsticks left.

How many matchsticks would you like to pick (1-4)? 1

There are 4 matchsticks left.

Computer picks 3 matchstick(s).

There are 1 matchstick left.

You were forced to pick the last matchstick. You lose!

**Output :**

You lose!