Structure Programming Language Laboratory

Assignment

<table>
<tr>
<td>1</td>
<td>WAP to take two string inputs from the user-first one is the main string, and the latter is the substring. Then, find if the substring occurred in the main string.
You cannot use built-in functions other than strlen().
<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>Mainstring: appleisagoodfruitappleisgood
Substring: apple</td><td>Found</td></tr>
<tr><td>Mainstring: Todayissaturday
Substring: monday</td><td>Not Found</td></tr>
</table>
</td>
</tr>
<tr>
<td>2</td>
<td>In the mystical land of Digitaria, a "Perfect Square Sorcerer" is a number that possesses an enchanting quality: it's a perfect square and its digits are arranged in non-decreasing order. These magical numbers, such as 16, 112225, and 1444, hold a captivating spell that fascinates mathematicians.

Your quest is to write a program that reveals all the hidden wonders of Perfect Square Sorcerers within a given range. Equip yourself with the following components to embark on this numerical adventure:

a. int is_perfect_square(int x): This function takes an integer x as input and returns 1 (or true) if it's a perfect square, and 0 otherwise.

b. int are_digits_non_decreasing(int x,int prev_digit): This function takes an integer x as input and returns 1 (or true) if its digits are arranged in non-decreasing order, and 0 otherwise.You must write this function using recursion.

c. int check_are_digits_non_decreasing(int x): This is a helper function which will start the recursive call by calling the function are_digits_non_decreasing()

d. void find_perfect_square_sorcerers(int start, int end): This function prints all the Perfect Square Sorcerers that lie within the range [start, end].
<table>
<tr><th>Samp[le Input</th><th>Sample Output</th></tr>
<tr><td>Enter the range [start, end]: 1 400</td><td>Perfect Square Sorcerers within the range [1, 400]:
1
4
9</td></tr>
</table>
</td>
</tr>
</table>

		16 25 36 49 144 169 225 256 289
3	You've been assigned the task of developing a program to manage employee records for a company. Each employee's record should include details such as their employee ID, name, age. The program should allow users to add a new employee, update employee information and display details of all employees. Tasks: 1. Create a structure named Employee which can store the following information: ● Employee ID (an integer) ● Name (a string) ● Age (an integer) 2. void addEmployee(struct Employee *EmployeeList, int *numEmployees) which allows the management to add employees to the system if the number of employees does not exceed 100. If an employee can be added then It should update the EmployeeList and increment the value of numEmployees. 3. void displayEmployee(struct Employee *Employee_ptr) which prints the details of the employee Employee_ptr is pointing to. 4. void updateEmployeeInfo(struct Employee *EmployeeList, int *numEmployees, char *EmployeeName) allows a user to update **Age** of a particular employee using the employee name. If no such employee exists, add the information as a new employee. In the **main** function: ● Create an array of Employees. [There can be 100 employees at max] ● Create an int variable numEmployees which will store the number of employees present in the company. ● Provide a menu for management to add employees (using addEmployee(-) function), update employee information and show the updated information (using updateEmployeeInfo(-) and displayEmployee(-) function). Sample Input/Output 1. Add Employee 2. Update Employee Information 3. Display All Employees 4. Exit	

Enter your choice: 1

Enter details for the new employee:

Employee ID: 1

Name: Sakib

Age: 26

New employee added.

1. Add Employee

2. Update Employee Information

3. Display All Employees

4. Exit

Enter your choice: 2

Enter the name of the employee to update: Sakib

Enter new age: 27

Employee information updated.

1. Add Employee

2. Update Employee Information

3. Display All Employees

4. Exit

Enter your choice: 3

Employee List:

Employee 1:

Employee ID: 1

Name: Sakib

Age: 27

4

In the bustling metropolis of Primeville, a **"Superhero Palindromic Prime"** is a prime number with a secret power: it remains unchanged even when its digits are reversed, just like the identity of a true superhero. For instance, numbers like **131 and 757** possess this extraordinary ability.

Your mission is to create a program that identifies and unveils all the hidden Superhero Palindromic Prime numbers within a given range. Channel your inner superhero and construct your code using the following components:

a. **int is_prime(int x):** This function serves as your prime power detector, taking an integer x as input and returning 1 (or a true value) if x is prime, and 0 otherwise.

b. **int reverse_number(int x):** This function possesses the power of time manipulation, accepting an integer x and returning the number obtained by reversing the order of its digits. **You must write this function using recursion**.

c. **int is_palindromic_prime(int x):** This function takes an integer x as input and returns 1 (or true) if x is a palindromic prime number (retains its form when its digits are reversed and also prime), and 0 otherwise. **You should make function calls to functions (a) and (b) in this function.**

d. **void find_superhero_palindromic_primes(int start, int end):** This function prints all Superhero Palindromic Prime numbers hidden within the range **[start, end].**

Sample Input	Sample Output
Enter lower limit: 10 **Enter upper limit: 400**	**Palindromic prime numbers within the range 10 to 1000 are:** **11** **101** **131** **151** **181** **191** **313** **353** **373** **383**

5

Tanjiro Kamado wants to eliminate as many demons as possible to make a better world for us. Every demon in our world can be presented as **an element of an array of structure**. Like,

```
struct demons {
    char name[60]; // Name of a demon int power; //
    Power of a demon
};
```

But he needs to fulfill two conditions before eliminating a demon. The conditions are:

a. **A demon has power multiple of 5, and**
b. **A demon's name contains only English alphabets.**

Tanjiro can only eliminate a demon who has power multiple of 5 and who has only English alphabets in his name. Like, if a demon has "Kute" as a name and power as 10 then Tanjiro can eliminate that demon. But if a demon has a name "Ku+e" as a name or power as 9 then Tanjiro can not eliminate that demon.

So your task is to find how many demons our Tanjiro can eliminate. Your implementation must have these two functions:

a) **int onlyAlphabets(char *input)** : This function returns 1 if the demon's name has only alphabets as letters (they can be capital or small letters) or return 0 otherwise.

b) **int Multiple_of_5(int x)** : This function returns 1 if the demon's power is multiple of 5. Returns 0 otherwise.

Sample Input	**Sample Output**
5 Kute 15 Ku+e 6 sMiley 10 ??x?? 35 PenTagoN 501	2