

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
HYDERABAD
CAMPUS,
Data Structures and Algorithms
CS F211 / IS F211

Homework Assignment - 2

1. Write a user interface driven C program to validate credit card numbers.

Note: Credit card numbers are validated using following method (called Luhn test)

The Luhn Formula:

1. Drop the last digit from the number. The last digit is called **check digit**.
2. Reverse the numbers
3. Multiply the digits in odd positions by 2 (if result is greater than 9, sum the digits)
4. Add all the numbers together
5. The check digit (the last number of the card) is the amount that you would need to add to get a multiple of 10 (Modulo 10)

For example: 30130708434187

1. 3 0 1 3 0 7 0 8 4 3 4 1 8
2. 8 1 4 3 4 8 0 7 0 3 1 0 3
3. 16 1 8 3 8 8 0 7 0 3 2 0 6 (7 1 8 3 8 8 0 7 0 3 2 0 6)
4. 53
5. $53+7 = 60$ (credit card number is valid)

2. Array based representation of sparse matrix and its transpose.

As you know there is no precise definition of when a matrix is sparse and when it is not, you will make the following assumptions about a sparse matrix. The matrix you consider for this exercise is of size 6 x 6. If it has less than 10 non-zero elements then you call it a sparse matrix.

Write a user interface driven C program to read a 6x6 matrix, create sparse representation (it will have 3 columns see figure below. 1st col represents row, 2nd col represent column, 3rd represent value) if required, transpose the sparse representation. **For example:**

	col 0	col 1	col 2	col 3	col 4	col 5
row 0	15	0	0	22	0	-15
row 1	0	11	3	0	0	0
row 2	0	0	0	-6	0	0
row 3	0	0	0	0	0	0
row 4	91	0	0	0	0	0
row 5	0	0	28	0	0	0

Only 8 out of 36 possible elements are nonzero. Hence, sparse!

Sparse matrix representation: <row, column, value> tuple (row is ordered)

	<i>row</i>	<i>col</i>	<i>value</i>
Array[0]	0	0	15
[1]	0	3	22
[2]	0	5	-15
[3]	1	1	11
[4]	1	2	3
[5]	2	3	-6
[6]	4	0	91
[7]	5	2	28

Transposed matrix: <row, column, value> tuple (it should be produced from your sparse representation)

	<i>row</i>	<i>col</i>	<i>value</i>
Array[0]	0	0	15
[1]	0	4	91
[2]	1	1	11
[3]	2	1	3
[4]	2	5	28
[5]	3	0	22
[6]	3	2	-6
[7]	5	0	-15

3. Point of sale (POS) simulation using structures.

Items in a departmental store can be modeled as a structure. Hence, you can write a C program using the structure to simulate the process of a departmental store.

For example, the structure that defines one item in a departmental store is as follows:

struct Item

```
{  
    char itemName[20];  
    int itemCode;  
    float price;  
    int QtyInStock;  
    int IsHighDemand;  
    int SoldToday;  
}Item;
```

Implement the following operations for the POS application.

1. Add a new item
 2. Update the price of an item
 3. Update the stock
 4. Show the price list
 5. Sell an item
 6. Exit
4. Write a user interface driven C program to implement a “railfence cipher”. A transposition cipher is one in which plaintext symbols are rearranged (i.e., transposed or permuted) to produce ciphertext

(encrypted text). The method of transposition may be either mathematical or typographical in nature. One such type of cipher is “railfence cipher”.

In railfence cipher, the encryption key is a positive integer.

Example: Suppose we want to encrypt the message “NOTHING IS AS IT SEEMS” with the encryption key (depth of rail fence) equal to 2.

Arrange the plaintext characters in an array with 2 rows (the key determines the number of rows) as shown below. Your program must support any key length encryption.

N T I G S S T E M

O H N I A I S E S (ignore space characters)

The ciphertext is produced by transcribing the first row followed by the second row.

Ciphertext: NTIGSSTEMOHNIAISES. So, you need to read plain text as well as key.

5. You all know the fundamental theorem of arithmetic also called as unique factorization theorem. So you are given a number N , do the prime factorization so that $N = p_1^{x_1} * p_2^{x_2} * ... * p_k^{x_k}$. You have find the sum of the exponents i.e. $(x_1 + x_2 + ... + x_k)$. **[Constraints: $2 \leq N \leq 10^8$. Time limit : 3s]**

Sample Input:

12

Sample Output:

3

6. Suppose you are inhabitant of a planet where 1, 7, and 9 are lucky digits. A lucky number for you is a number that contains only your lucky digits in it. For ex: 1, 79, 911, 9917 etc., are lucky, where as 5, 172, 93, 170 are not. Given a integer N , count the number of lucky numbers in the range 1 to N . **[Constraints: $1 \leq N \leq 10^{12}$. Time limit : 3s]**

Sample Input:

71

Sample Output:

7

Explanation : The lucky numbers that are not more than 71 are 1, 7, 9, 11, 17, 19, 71

7. You are caught by math’s geeks and they are not leaving you asking to solve a problem. You need to save your life by solving the problem. They give you integer N and K and ask you to find the smallest and the largest positive integers each containing exactly N digits and having exactly K distinct digits from 0 to 9. For ex: 474545 has only 3 distinct digits 4, 5 and 7. Obviously leading 0’s are not allowed, for ex: 0145 is not allowed. You will get the input in only one line, containing N K ($1 \leq N \leq 18$ and $1 \leq K \leq 10$). Print the smallest and largest integer.[Input is given such that the answer will always exist]

Sample Input:

3 3

Sample Output:

102 987

8. Most of the people don’t like comments. A C program is inspired by such thinking and ask you a favor to remove all comments from it. You have to read a C program file (use fopen) and then output

the program on your console removing all comments.

Sample Input :

```
#include<stdio.h>
/* Author : XYZ
 * Date : 21/1/2016
 */
int main()
{
    int a; // variable a
    return 0;
}
```

Sample Output:

```
#include<stdio.h>
int main()
{
    int a;
    return 0;
}
```

9. You are given a sorted array of names (only first name) and a name **X** for which you need to find out how many times it is present in the array. You have to only use binary search to solve this problem. (No linear search). Also you cannot use any string library function.

Sample Input:

```
3
abc
pqr
abc
pqr
```

Sample Output:

```
1
```

Explanation: **X** = "pqr" is present only once hence output is 1

10. A positive integer is called a **palindrome** if its same when read from left to right and from right to left. If you are given a positive integer **X** (of not more than 1000 digits, it can be large...), write the value of the smallest palindrome larger than **X**. Numbers are always displayed without leading zeros. (**Note :** You should not increase **X** by 1 and check if it's palindrome)

You have to read first **N** which is total number of integers to read. And then following **N** lines will contain **N** numbers for which you need to find out immediate next palindrome.

Sample Input:

4
5
11
515
1234

Sample Output:

6
22
525
1331
