PSG COLLEGE OF TECHNOLOGY
DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCES
OBJECT COMPUTING LAB
WORKSHEET (C Programming – Revision)

CONTROL FLOW STATEMENTS

ws_p1. In mathematics, a Mersenne prime is a prime number that is one less than a power of two. That is, it is a prime number that can be written in the form

$$M_n = 2^n - 1 \text{ for some integer } n$$

Write a C program to find and prints the smallest 5 Mersenne primes.

ws_p2. **SMITH NUMBER**: It is a composite integer with the property that the sum of its digits is the same as the sum of the digits of its prime factors. A composite integer N whose digit sum S(N) is equal to the sum of the digits of its prime factors Sp (N) is called a Smith number.

For example 85 is a Smith number because digit sum of 85 i.e. S(85) = 8 + 5=13, which is equal to the sum of the digits of its prime factors i.e. Sp (85) = Sp (17 x 5) = 1 + 7 + 5 = 3.

Test Data: 4, 22, 58, 85, 94, 121, 166, 202, 265, 274

Write a C program to read a composite integer and check whether it is a smith number or not.

ws_p3. **FRIENDLY NUMBERS**: In number theory, a friendly number is a natural number that shares a certain characteristic called abundancy, the ratio between the sum of divisors of the number and the number itself, with one or more other numbers. Two numbers with the same abundancy form a friendly pair.

For example the friendly pair (6, 28) with abundancy

$\sigma(6) / 6 = (1+2+3+6) / 6 = 2,$

$\sigma(28) / 28 = (1+2+4+7+14+28)/ 28 = 2.$

The shared value 2 is an integer in this case but not in many other cases.

Some of the friendly pairs are: (6, 28), (30, 140), (80, 200), (210, 224), (12, 234), (66, 308).

Write a C program to read two natural numbers and check whether they are friends or not.

ws_p4. Write a C program to perform the following test to find the given number is divisible by 7.

Read an integer, remove the last digit, double it, subtract it from the truncated original number and continue doing this until only one digit remains. If this is 0 or 7, then the original number is divisible by 7.

Example: 1603 -> 160-2(3)=154 -> 15-2(4)=7, so 1603 is divisible by 7.

ws_p5. Write a C program to perform the following test to find the given number is divisible by 11.

Take the alternating sum of the digits in the number, read from left to right. If that is divisible by 11, so is the original number.

So, for instance, 2728 has alternating sum of digits 2-7+2-8 = -11. Since -11 is divisible by 11, so is 2728.

For 31415, the alternating sum of digits is 3-1+4-1+5 = 10. This is not divisible by 11, so neither is 31415.

ARRAYS

ws_p6. Write a C program to cyclically rotate an array by 'r' times.

Example, given an array, cyclically rotate the array by '2' times (here, r=2)

Input: arr[] = {1, 2, 3, 4, 5}

Output: arr[] = {4, 5, 1, 2, 3}

ws_p7. Write a C program to find the Union and Intersection of two sorted arrays.

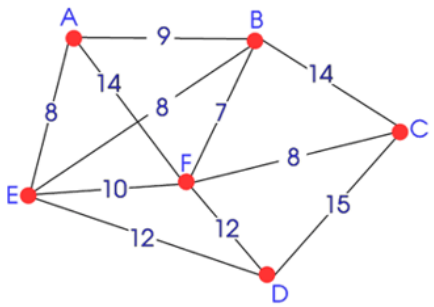
For example, if the input arrays are: arr1[] = {1, 3, 4, 5, 7}, arr2[] = {2, 3, 5, 6}

Then your program should print Union as {1, 2, 3, 4, 5, 6, 7} and Intersection as {3, 5}.

ws_p8. In numerical analysis, a sparse matrix is a matrix in which most of the elements are zero. Here, substantial memory requirement reductions can be realized by storing only the non-zero entries. One such format is Coordinate List (CL), which stores sparse matrix as a list (row, column, value) tuples. Write a C function to accept a sparse matrix and print its CL format. For example, consider the following sparse matrix and its Coordinate list format.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 9 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 7 & 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 4 & 9 \\ 1 & 1 & 8 \\ 2 & 0 & 4 \\ 2 & 3 & 2 \\ 3 & 5 & 5 \\ 4 & 2 & 7 \\ 4 & 5 & 1 \end{bmatrix}$$

ws_p9. Consider the following graph



	A	B	C	D	E	F
A	0	9	0	0	8	14
B	9	0	14	0	0	7
C	0	14	0	15	0	8
D	0	0	15	0	12	12
E	8	0	0	12	0	10
F	14	7	8	12	10	0

The letters in the diagram represent towns. The numbers show the distance between those towns in km. One way to store this graph in a computer program is to use an adjacency matrix which is a two dimensional array. The matrix is N × N where N is the number of nodes. At each location in the matrix is stored the distance of the edge that connects those two towns if there is an edge. For the locations in the adjacency matrix where there is no edge, we can store a sentinel value such as 0 or -1.

Write a C function '`funDistance(int adj[][], int tour[])`' to calculate and return the total distance for a tour. This function accepts two parameters: the adjacency matrix and a tour matrix. The tour should start and end at the same town, and it should visit all the towns exactly once. The function should return zero if the tour is invalid.

For an example tour [A-E-F-D-C-B-A] or {1, 5, 6, 4, 3, 2, 1}, the cost is → 68

ws_p10. Write a C program to solve the following situation. Allen and Bruce are working as a salesman for a cookie company. Every day they have to visit to the nearest cities to promote the sales. At the end of the day both of them will submit a list of cities that they visited on that day. From their list, the sales manager wants to find the cities which are uncommon from Alice & Bruce. Consider the following assumptions:

The cities are numbered from 1 to 20,

The cities could be visited in any order, and

The number of cities visited might be different from Allen & Bruce.

For example, Allen visited the cities {4, 6, 1, 10, 7} and Bruce had a list {1, 5, 4, 11, 15, 3}

The uncommon cities from Alice are {6,10,7} and the uncommon cities from Bruce are {5,11,15,3}

STRINGS

ws_p11. Write a C function to find maximum depth of nested parenthesis in a string. For example, consider a string having parenthesis like below

"(((A)) (((B))))"

You need to find the maximum depth of balanced parenthesis, like 4 in above example (Since 'B' is surrounded by 4 balanced parenthesis). If parenthesis are unbalanced then return -1.

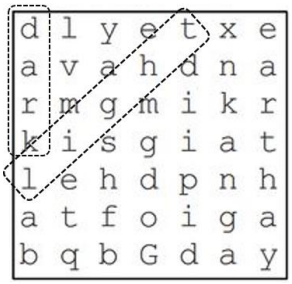
ws_p12. Define a function to compare two sub-strings from a specified starting index of two different strings. The function should return 1 when both the substrings are equal, 0 otherwise. For example,

- If S1 = "STEERING" & S2 = "SPRING",

Comparing the sub-strings with the index from 4 for S1 and 2 for S2, returns 1 as they are equal. (The substrings are: from S1 is "RING" and from S2 is "RING").

ws_p13. Write a C function to compare two strings in terms of alphabets and their frequencies rather than with their corresponding indices. The function should return 1 if both are equal, 0 otherwise. And demonstrate the function with possible inputs. For example, “THERE” and “THREE” are equal as they have same alphabets and their count is also same.

ws_p14. There is a 2D grid of characters. You need to find if a given word can be matched in any direction in the grid. The direction can be up-down, down-up, left-right, right-left, and both diagonals up and down.



ws_p15. You are given a 2D array of characters and a character pattern. Write a C program to find if pattern is present in 2D array. Pattern can be in any way (all 8 neighbours to be considered) but you can’t use same character twice while matching. Return 1 if match is found, 0 if not. For example, the pattern “MICROSOFT” is marked in the given array.

N	F	N	K	D
N	E	C	I	L
A	R	E	K	M
P	O	S	F	T
X	S	O	T	I

STRUCTURES

ws_p16. Define a structure to store customer details as given below.

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3005	Graham Zusi	California	200	5002
3001	Brad Guzan	London	100	5005
3004	Fabian Johns	Paris	300	5006
3007	Brad Davis	New York	200	5001
3009	Geoff Camero	Berlin	100	5003
3008	Julian Green	London	300	5002
3003	Jozy Altidor	Moncow	200	5007

Write a program to read the customer details from and functions to achieve the following.

- Display the customer’s detail who is living in a particular city.
- Display the customer’s name whose name starts with a particular alphabet.
- Display the salesmen id, who are all having more than one customer.

ws_p17. Consider the following structure declaration to store a 2D coordinate point.

struct **Point** { int x, y; };

Declare another structure to store a **Rectangle** with its top-left and bottom-right coordinate points (make use of the **Point** structure to store the points). Provide an implementation to read the information about N rectangles, and sort them based on their area and print them. Also write the functions to perform the following test.

- Read a point and check whether it is inside any of the N rectangles, and return the number of rectangles for which the point is inside.
- Compare two adjacent rectangles and return 1 when they are equal in dimensions, 0 otherwise.
- Compare two adjacent rectangles and return 1 if the first rectangle is completely contained inside the second rectangle, return 2 if the first rectangle is completely outside, and return 3 when they are overlapping each other.

FILES

ws_p18. Write a C program to read a 2-dimensional matrix of size 10×5 and store it into a file called 'twodimension.txt'. Further, read the matrix from the file and store it in row-major order and column-major order into the files 'rowmajor.txt' and 'colmajor.txt' respectively.

ws_p19. Consider the following two tables, write a program to define corresponding structures, read & store the details into the files "salesman.txt" and "customers.txt" respectively.

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5003	Lauson Hen	Berlin	0.24
5004	Jack	Rome	0.11
5005	Pit Alex	London	0.21
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3005	Graham Zusi	California	200	5002
3001	Brad Guzan	London	100	5005
3004	Fabian Johns	Paris	300	5006
3007	Brad Davis	New York	200	5001
3009	Geoff Camero	Berlin	100	5003
3008	Julian Green	London	300	5002
3003	Jozy Altidor	Moncow	200	5007

Write another program to read both the customer and salesman details from the text files and write the code for the following.

- Display those customers with their name and those salesmen with their name and city who lives in the same city.
- Display the salesman who is getting more than 15% commission.
- Display the salesman who doesn't have any customers.

ws_p20. **PIGGY BANK –** Piggy Bank is a small finance bank, which offers a customer to open a savings account and do transactions like cash deposit, cash withdrawal, and fund transfer. The detailed transactions are given below

- Initially a customer open an account with customer name and a 4-digit pin number, the account number will be provided by the bank (4-digit number starts with 1001).
- After account registration, a customer should logon to his account with his account number and pin number to perform any of the following transactions.
  - Cash deposit – the customer should enter an amount to be deposited.
  - Cash withdrawal – the customer should enter an amount to be withdrawn.
  - Fund transfer – the customer should enter the account number of the receiver and the amount to be transferred.
  - Mini statement – the customer can view their last 3 transactions.
  - Pin change – the customer should be able to change his pin number.
- All the above transactions has to be carried out with appropriate conditions, for example, a customer can't withdraw more than what he has.

Write a C program to implement "**Piggy Bank**". The implementation must have at least the following tables stored in 'Customers.txt' and 'Transactions.txt'. [Hint – the data in the tables are given as a sample]

Customers.txt

Account No.	Customer Name	Pin	Balance
1001	Mougli	1122	20500
1002	Baloo	2017	12600
1003	Bagheera	1607	7500

Transactions.txt

Account No.	Type of Transaction	Amount	Transferred To
1001	Deposit	500	
1003	Withdrawal	1000	
1002	Transfer	1000	1003
1003	Withdrawal	5000	
1002	Deposit	1000	
1001	Withdrawal	2000	
1001	Transfer	1000	1003

ws_p21. **Bits & Bytes [BB] –** BB is a software development company. Currently they wanted to develop an in-house application to track their work flow based on the following scenario. This application involves three facts: Developers, Projects, and Teams. The application should allow us to do the following manipulations

- Should be able to add or delete developers / projects / teams
- Should be able to view the list of different details about developers / projects / teams

Develop a C program to design a work flow system as described above. The system should maintain the following files to maintain developers, projects, and teams' detail. [Hint – the data in the tables are given as a sample]

Developers.txt

Developer Id	Name	Team Id	Project Id
D01	Sparrow	T03	P01
D02	Ragetti	T01	P02
D03	Anamaria	T02	P03

Projects.txt

Project Id	Project Title	Client Id	Cost
P01	Blog	C01	10000
P02	Mobile-App	C02	25000
P03	Mail-Box	C03	35000

Teams.txt

Team Id	Division	Manager Name
T01	Scripting	Barbossa
T02	Database	Gibbs
T03	Testing	Elizabeth

These tables can be updated by specifying sequence of operations in a separate file called 'Query.txt'. The operations should follow the given syntax. (The bold face are keywords)

- To insert a record to developer the query should be - **INSERT** D01,David,T01,P02 **INTO** Developer
- To insert a record to project – **INSERT** P01,Blog,C01,10000 **INTO** Projects
- To delete a record from project – **DELETE** P01 **FROM** Projects
- To view the Team details – **DISPLAY** Teams
- To view the list of developers working in project 'P01' – **DISPLAY** Developers **WITH** P01
- To view the division and manager name for the developer 'D02' – **DISPLAY** Team **FOR** D02

ws_p22. Opinion mining is a type of natural language processing for tracking the mood of the public about a particular product. Opinion mining can be useful in several ways. It can help marketers evaluate the success of an ad campaign or new product launch, determine which versions of a product or service are popular and identify which demographics like or dislike particular product features. For example, a review on a website might be broadly positive about a digital camera, but be specifically negative about how heavy it is. Being able to identify this kind of information in a systematic way gives the vendor a much clearer picture of public opinion than surveys or focus groups do, because the data is created by the customer. In opinion mining procedures, the stop word removal and word stemming are the initial steps to be carried out to improve the quality of opinion mining. For example consider the following review from a customer

"I order[ed] this product through amazon prime on 21st and it arriv[ed] on 23rd. On unbox[ing] I found there is a manufactur[ing] defect with the screen which show[s] the quality check on their part. There were vertical line[s] on the screen display. For everyone who's plann[ing] to buy this product, know that the quality check has deteriorat[ed] for OnePlus. The issue was clear[ly] visible from the point I turn[ed] the phone on. And since there is a stock shortage, be ready to face anything."

The underlined are stop words, and the boxed alphabets represent stemming to reach the root word.

Based on the above scenario, assume that you have given a text file 'opinions.txt', contains a collection of user reviews. Write a C program to remove the following stop words

{“this”, “and”, “it”, “on”, “the”, “is”, “to”, “be”, “for”, “at”}

and stem the words which ends with ‘s’, 'ed', 'ing' and 'ly'.