Mini Project
3rd Semester

Project 1:-
Tic Tac Toe

Tic-tac-toe is a paper-and-pencil game for two players, **X** and **O**, who take turns marking the spaces in a 3 × 3 grid. The game may be started either by the **X** player or by the **O** player. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game. For example, Figure 1 shows a tic-tac-toe game where player **O** wins through a diagonal line. However, it is not necessarily the case that every game has a winner (e.g., see Figure 2): when there is not a single free slot in the grid, and when neither player has managed to mark a winning row, then it is called a tie. That is, a game is considered as over either when there is a winner (e.g., Figure 1), or when there is a tie (e.g., Figure 2).

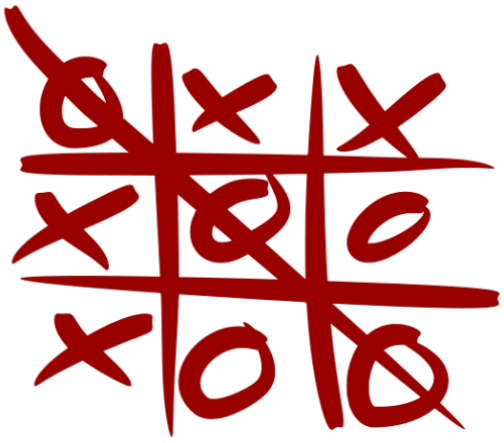


Figure 1: Tic-Tac-Toe: The O Player Wins

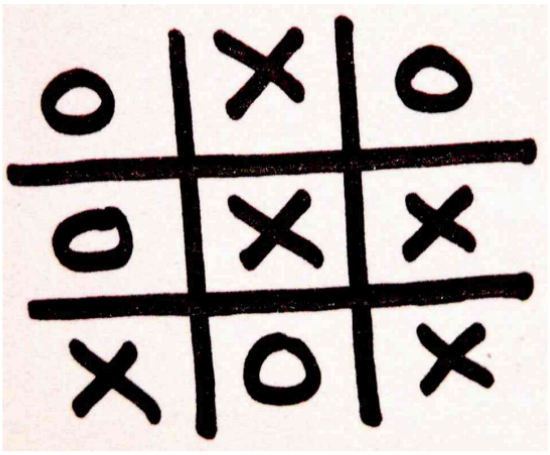


Figure 2: Tic-Tac-Toe: A Tie

Your Task

In this assignment you are required to design a tic-tac-toe game. Your program must exhibit the following behaviour:-

1) Prompt the user for the name of the X player, then the name of the O player. For example, the user may enter Akshay as the X player and Ashish as the O player:
2) Ask the user who will play first in the current game. Your program must give an hint on names

```
Enter a name for the X player:
Akshay
Enter a name for the O player:
Ashish
```

of the two registered players (e.g., Akshay and Ashish). If the user enters a name that does not match the name of either the X player or the O player, then print an error message. For example:

```
Who plays first, Akshay or Ashish?
Akash
Akash is not a registered player.
Who plays first, Akshay or Ashish?
Akshay
```

Notice that in the prompt message, Akshay and Ashish are given as an hint as to what the user is expected to type. Your program must output such a hint. If necessary, you must repeatedly prompt the user to re-enter a name until it matches one of the names that they entered.

3) Once the user specifies a valid name for the player to play first, your program must print out the initial state of the grid, where all nine slots in the grid are unoccupied. We use . (i.e., a dot) to represent an unoccupied slot. Also, your program must print out the name of the player for the current turn. For example:

```
...
...
...
Player of current turn: Akshay
```

4) For each turn of the game, your program will:
   1) Prompt the user to first specify an integer row number. If the row number is not between 0 and 2, then you program must print an error message, and repeatedly ask the user to re-enter a new row number until it is in the correct range.
   2) Prompt the user to specify an integer column number. If the column number is not between 0 and 2, then you program must print an error message, and start over again by asking the user to enter the row number and then the column number.

```
Choose a row number (0 to 2):
3
3 is not a valid row.
Choose a row number (0 to 2):
2
Choose a column number (0 to 2):
-1
2
-1 is not a valid column.
Choose a row number (0 to 2):
2
Choose a column number (0 to 2):
1
. . .
. . .
. X .
```

3) Check if the slot in the grid, identified by the user-entered row and column numbers, is already occupied (i.e., marked by an X or an O). If the identified slot is already occupied, then your program must print an error message, and start over again by asking the user to enter the row number and then the column number.
That is, your program must repeat Steps 4.1 to 4.3 until the slot identified by the user entered row and column numbers is free, and can thus be placed a mark.
4) Once the user-entered row and column numbers pass these checks, depending on if the player for the current is an X player or an O player, your program must change the state of the grid and output the updated state. Use X (capital letter X) for the mark of the X player, and O (capital letter O) for the O player. For example:

Remember that in our running example, Akshay, who is an X player, is the player of the current turn, so after the row number 2 and column number 1 pass the checks, an X mark is placed in that slot.
Your program will repeat Step 4 until either: 1) there is a winner; or 2) there is a tie. When either of the above occurs, your program must print an appropriate message to conclude the current game, and ask if the user would like to start a new game. The user's answer must be either Y (for yes) or N (for no), and if necessary, your program must repeatedly prompt the user for a new answer until a valid answer is entered. As an example of Akshay being the player of the current turn and winning the game:

```
Game is over:
Akshay wins!
Would you like to play again? (Y/N)
```

As an example of Akshay being the player of the current turn and placing a mark on the last available slot without winning:

```
Game is over:
it is a tie!
Would you like to play again? (Y/N)
Yes
Yes is not a valid answer.
Would you like to play again? (Y/N)
Y
Enter a name for the X player:
```

Actually, if the result is a tie, then the player of the ending turn must be whoever started first (e.g., Akshay started first in our running example). **Why?**

5) Once the user decides not to play again, your program must terminate and say goodbye to them. For example:

```
Would you like to play again? (Y/N)
N
Bye!
```

Project 2:-
Random password generation:-
Write a program to generate random password which have the following features:-

1) The password should have minimum length of 12 characters and maximum length of 32 characters.
2) Password will always start with a lower case alphabet and ends with a uppercase alphabet.
3) It should have at least 2 lower case alphabet, 2 upper case alphabet, 1 number, and 1 special character. No space allowed in generated password.
4) You should not use any dictionary for password generation.
5) Do not use inbuilt function for randomisation make your own pseudo-random number generator.
6) GUI - There should be one button for generating password which will be shown on same screen inside some label, there should be one button which will copy the password on the clipboard.

Project 3:-
File Encryption:-
Write a program that takes a file as command line input and output a file that is in encrypted form, take the following features in mind:-

1) File to encrypt can be of any type. For ex. if the decrypted file is abc.mp4 the encrypted file will be abc.mp4.ecrypt.
2) Your function should be able to encrypt as well as decrypt the file.
3) Your function should ask for a key to decrypt the file.
4) You should not use any pre-built library to encrypt/decrypt file, instead make your own function.
5) Your file should not decrypt using any password other that given password.
6) File can be of max. 1GB.
7) While encrypting, your program should show progress.
8) You can read this paper
   http://www.iaeme.com/MasterAdmin/UploadFolder/IMAGE%20ENCRYPTION%20AND%20DECRYPTION%20USING%20AES%20ALGORITHM/IMAGE%20ENCRYPTION%20AND%20DECRYPTION%20USING%20AES%20ALGORITHM.pdf
   to understand more about encryption using AES.

Project 4:-
Sudoku Easy (6 X 6):-
Write a program that generates a random (easy) 6x6 sudoku puzzle in every run. There should be at least one row / column / block which contains 4 or more entries initially. The script should prompt the user to fill in the blanks with numbers from 1-6 and decide if it is completed successfully or not. Add a timer option in the above game, such that the user's score will reduce after timed out. The reduction of score should be directly proportional to the extra time user has taken.
1) The program should have a Desktop GUI (It can be web interface).

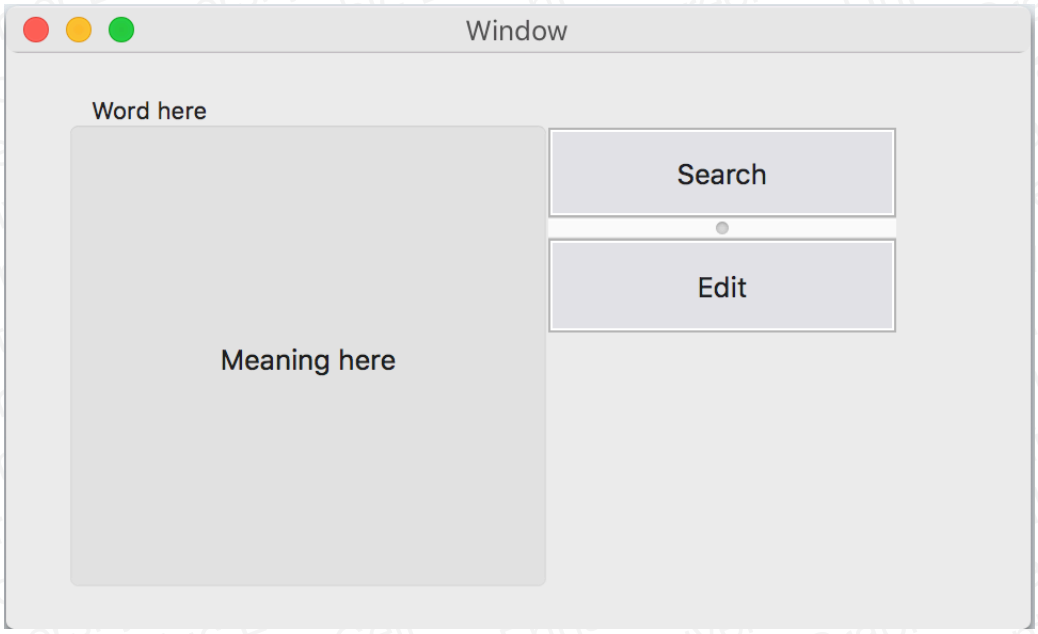
Project 5:-
Big Integer Representation:-
Write a program in C/C++ to implement Big Integers and perform calculation (big integer are those which do not have any maximum or minimum value they can be build until computer runs out of memory). Your program should be able to:-

1) Create big integer.
2) Add, subtract and multiply two big integers.
3) Integer division of two big integer.
4) Do not use addition for multiplication.

[NOTE:- negative numbers should be handled]

Project 6:-
Dictionary:-
Write a program to implement dictionary. Your program should have following features:-

1) You should use a text dictionary to be loaded into program at the time of starting and should write back after modification.
2) User should be able to search the meaning of a word in dictionary.
3) User should be able to add/ delete/ modify a word and its meaning.
4) Searching a meaning should be of constant time.
5) The program should have GUI, user should be able to enter some word in a text box and search it by pressing the search button, It should also have an `edit` button through which user should be able to modify the meaning of the searched word. The `add` button will allow the user to add a new word (the new word should not be available in the dictionary) and its meaning, and if it is already available the program should give error.



Project 7:-
File merging:-
Write a program that reads a sequence of file**s** from the disk that contain three fields <name> <age> <salary> separated by space in each row. Perform merge-sort of these files on the given field(s) (eg: age, or age name, or name age, or name age salary) and generate the output in a file. If two or more fields are given, sort on the first field and if two entries are same, sort on the second field (and so on).

1) It should have a GUI using which user can add files and it should give the output file based on given filters.

Project 8:-
Zip Password Crackers:
Write a simple brute force C/C++ program to generate permutations (possible passwords) which can crack the password protected zip file.
Read this for more details:-
terminal program
You should not use any system call to run the terminal commands. you can use this library.
Your program should have a dictionary of passwords on which it tries permutation. Suppose your dictionary has one word `abc` the program will check all 'abc', 'acb', 'bac', 'bca', 'cab', 'cba'.

Project 9:-
Dice-ware Password generator:-
Write a simple program to generate a password using dice ware password generation algorithm. suppose you have a dictionary

...
43136 mulct
43141 mule
43142 mull
43143 multi
43144 mum
43145 mummy
43146 munch
43151 mung
...
you roll a dice 5 times and you got 4,3,1,4,4 that means you have `mum` as your word. You will ask user how many words he wants in his password and program will generate that many words using the dictionary.
See:- (https://en.wikipedia.org/wiki/Diceware)
Check this out :- https://www.rempe.us/diceware/#eff
You can use this dictionary for password - http://world.std.com/~reinhold/dicewarewordlist.pdf

Project 10:-
Canteen Management System:-
Write a CLI (Command Line Interface) software to manage canteen of GEU. It should have following features:-
1) Menu driven.
2) Account System to manage the total purchase and sale. It should show the profit and loss.
3) It should be able to calculate the cost of items which are prepared at the canteen (Like Coffee). Suppose 1 cup of coffee takes 50 ml of milk, 5g sugar, 10g coffee powder, 1 cup than based on the per unit rate (1L milk rate, 1 kg sugar rate etc..) of these material your program should be able to tell the cost of the coffee.
4) It should be able to store different things on file so that user need not to do type cost, sale etc.. again when it starts the software again and again.
5) Should able to tell the sale of today, past week, past month and past year. Also the profit and loss of the respective time period.
6) It should be able to tell if a product is below threshold and out of stock to the canteen manager to buy the product.

Project 11:-
Library Management System:-
Write a CLI (Command Line Interface) software to manage library of GEU.It should have following features:-
1) Menu driven.
2) Record of all the books which are in the library.
3) Records of the books which are issued and return date should be automatically set to 30 days after current date (Note:- months have different number of days).
4) Student should be added to the database to issue books.
5) You can maintain some files to store database.
6) Library manager should be able to see the availability of any book. He should also be able to see which book is given to which student and which student took how many books along with book names.

Project 12:-
TF-IDF:-
In this project you will find the term frequency, inverse document frequency of different terms(words). This is very good project I recommend you to use python for this project as it contains very rich libraries and is easy to use.
[Read:- TF-IDF] for more information on TF-IDF. You are free to choose any document of your choice which should be pass as command line argument. The TF-IDF of each word which are passed as command line argument should be saved in a `output.txt` file.

```
./a.out document word1 word2 word3 …
```

Project 13:-
Spell-Corrector
Write a program in C, C++ or JAVA to correct the spelling of input word, for e.g, if the input word is `speling` the output word will be `spelling` or if the input word is `korected` the output word will be `corrected`. You can check Peter Norvig's explanation for spelling correction here or you can use Levenshtein Distance to implement your solution.
[NOTE:- Some code is provided in the above mentioned link but you are advised to code on your own without looking at the given code. Looking at the code might bias your thinking and ultimately you end up coding the same code which will be detected by our system].