Programming Fundamentals *COURSE & LAB*

Fall 2022

(BS-IT-F22 Morning)

Assignment # 2

Assigned on: ***Saturday, Dec 09, 2023***

*Submission Deadline:****Monday Dec 11,2023***
(till 11:30 PM)

Instructions:

- **This is an individual assignment. Absolutely NO collaboration is allowed. Any traces of plagiarism/cheating would result in an "F" grade in this course.**
- Do **NOT** copy even a single sentence/statement or line of code from any other person or book or Internet or any other source.
- This assignment will ALSO be counted towards your marks in **PF-Lab**.
- Late submissions will NOT be accepted, whatever your excuse may be.
- This assignment needs to be submitted in **Soft** form. The **Submission Procedure** is given below.

Note:

- Implement all the task as **functions**
- You are not allowed to take input in the main function and not in the function which performs a specific task for input make a separate function like if you want to take height as an input write function which has a prototype like this **int getHeight();** Perform all the input validation in this function.

Submission Procedure:

1. You must write **C code** (using the notations/conventions we have discussed) in a **single file**. Also make sure that you **write your Roll Number and Name at the top of your VSCode file.**
2. The name of your submission file must be exactly according to the format:
   Mor-A1-BITF22M032
3. where the text shown in **BLUE** should be your **complete Roll Number**.
4. Finally, upload and submit your file through **Google Classroom**. Make sure that you press the SUBMIT button after uploading your file.

Note: *If you do not follow the above submission procedure, your Assignment will NOT be graded and you will be awarded a ZERO.*

You are required to write a C program for the following tasks. Keep the following instructions in mind when writing your algorithms:

- Use meaningful variable names. Use ***camelCase*** notation to name variables.
- ***Indent*** the code properly.
- Use meaningful prompt lines for all input, and use meaningful labels for all output that is performed by your program.

1. String Tokenization (25 points)
Write a C function that takes a sentence and a delimiter as input and tokenizes the sentence into individual words. The function takes a 2D array and places individual words on separate indexes.

Explanation:

Input Sentence: "This is a sample sentence."
Delimiter: ' '

```
Tokenized Words:
- This                at index [0][0]
- is                  at index [1][1]
- a                   at index [2][2]
- sample              at index [3][3]
- sentence            at index [4][4]
```

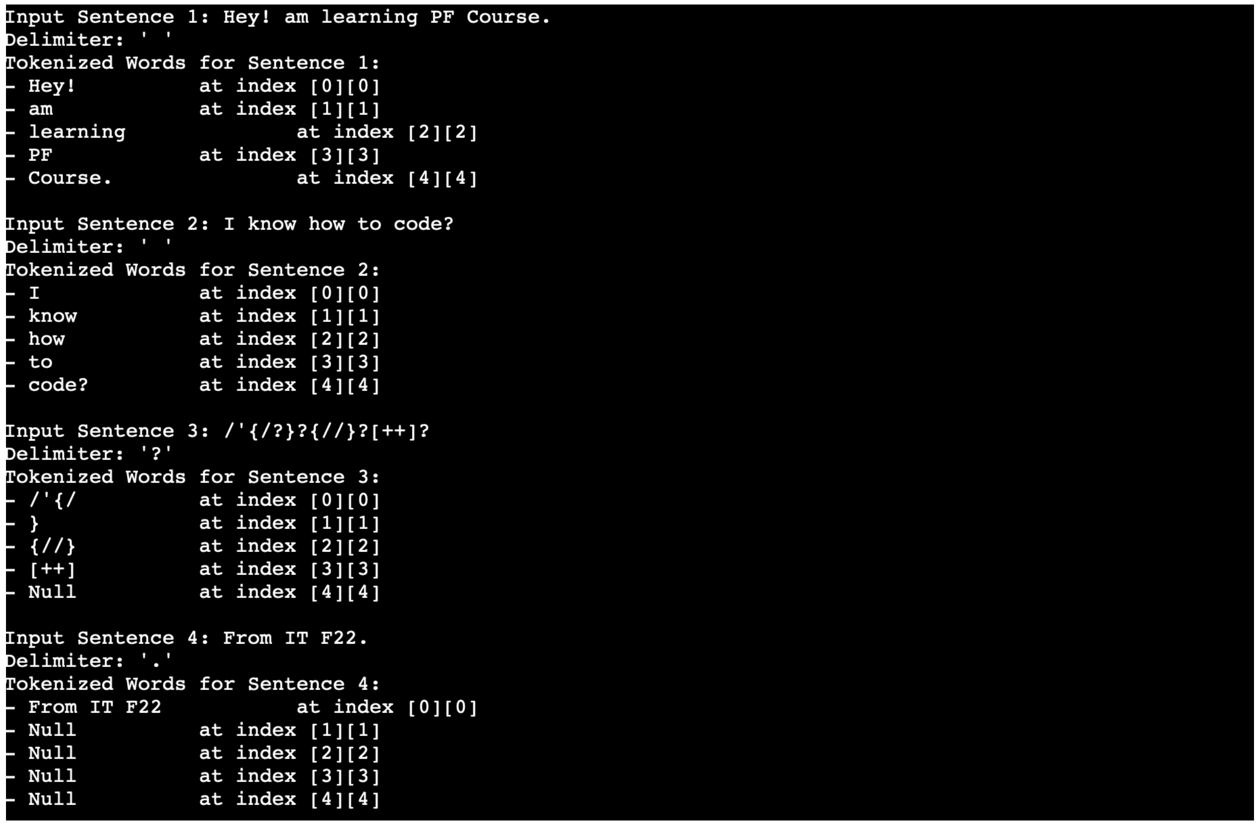
Input Sentence: Hello, world! How are you?
Delimiter: ,

```
Tokenized Words:
- Hello               at index [1][1]
- world! How are you? at index [1][1]
‘\0’                  at index [2][2]
‘\0’                  at index [3][3]
‘\0’                  at index [4][4]
```

Note: Assume the length of words in Input Sentence isn’t greater than 10 and the number of words in a sentence isn’t greater than 5.
Function Prototype:

Void takeSentence(char [], int size)	**(2)**
Void takeDelimiter(char [], int size)	**(1)**
Void printDelimiter(char del)	**(1)**
Void printSentence(char [],int size)	**(2)**
void tokenizeSentence(char sentence[ ], char delimiter, char tokens[ ][ ])	**(15)**
Void print2D(char tokens[ ][ ], int r,int c)	**(4)**

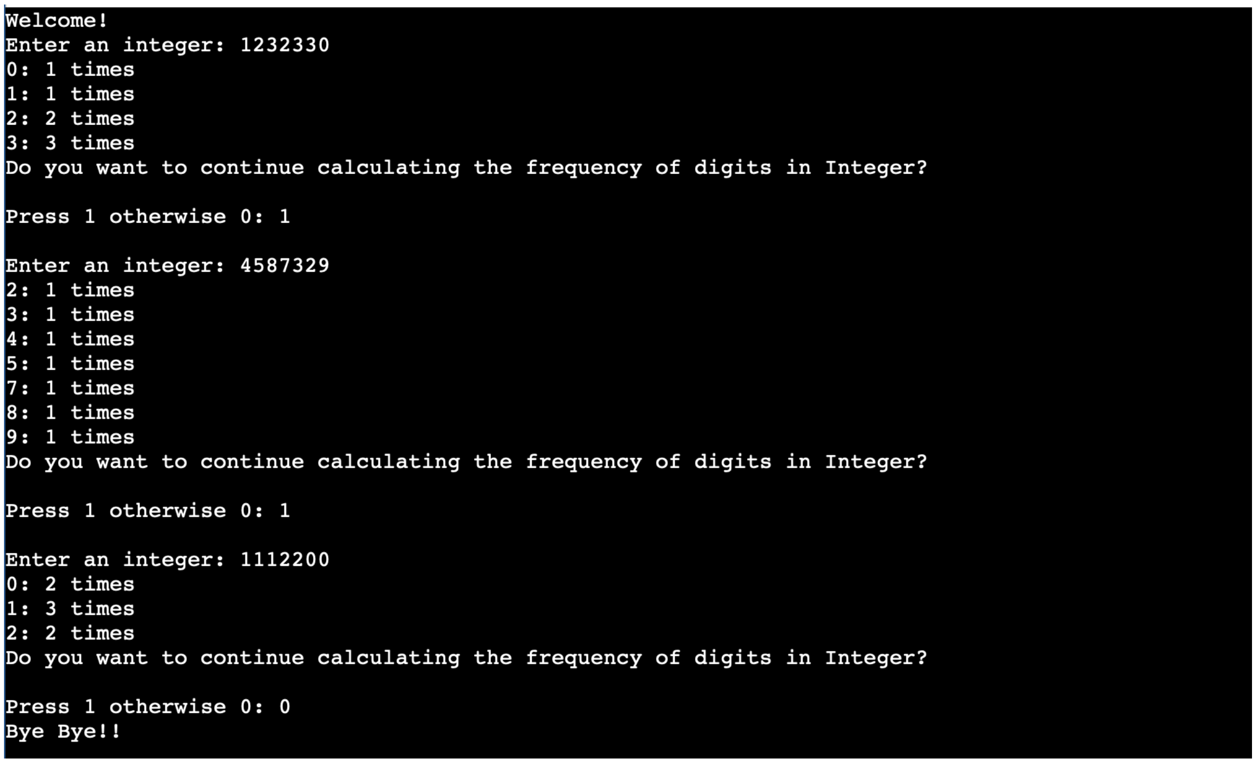
char tokens[ ][ ]-->Think about this 2D array size.



```
Input Sentence 1: Hey! am learning PF Course.
Delimiter: ' '
Tokenized Words for Sentence 1:
- Hey!          at index [0][0]
- am            at index [1][1]
- learning              at index [2][2]
- PF            at index [3][3]
- Course.               at index [4][4]

Input Sentence 2: I know how to code?
Delimiter: ' '
Tokenized Words for Sentence 2:
- I             at index [0][0]
- know          at index [1][1]
- how           at index [2][2]
- to            at index [3][3]
- code?         at index [4][4]

Input Sentence 3: /'{/?}?{//}?[++]?
Delimiter: '?'
Tokenized Words for Sentence 3:
- /'{/          at index [0][0]
- }             at index [1][1]
- {//}          at index [2][2]
- [++]          at index [3][3]
- Null          at index [4][4]

Input Sentence 4: From IT F22.
Delimiter: '.'
Tokenized Words for Sentence 4:
- From IT F22           at index [0][0]
- Null          at index [1][1]
- Null          at index [2][2]
- Null          at index [3][3]
- Null          at index [4][4]
```

Make Sure your program doesn't print Null.Print all indexes instead of those containing Null.

2. Digit Frequency (10 points)
Given an integer, find out the frequency of each digit in it.
Note: You are not allowed to use nested loops.
Also make sure to do input validation for the driver function as well.



```
Welcome!
Enter an integer: 1232330
0: 1 times
1: 1 times
2: 2 times
3: 3 times
Do you want to continue calculating the frequency of digits in Integer?

Press 1 otherwise 0: 1

Enter an integer: 4587329
2: 1 times
3: 1 times
4: 1 times
5: 1 times
7: 1 times
8: 1 times
9: 1 times
Do you want to continue calculating the frequency of digits in Integer?

Press 1 otherwise 0: 1

Enter an integer: 1112200
0: 2 times
1: 3 times
2: 2 times
Do you want to continue calculating the frequency of digits in Integer?

Press 1 otherwise 0: 0
Bye Bye!!
```

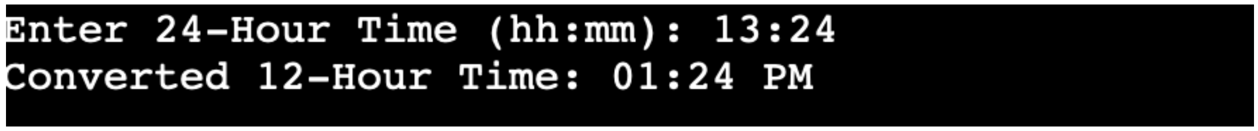
3. Calculation of Age(5 points)

Write C program which take age as integer and convert into

- Age in Months
- Age in Hours
- Age in Minutes
- Age in Seconds

4. Conversion of Time(5 points)

C Program to convert 24 Hour time to 12 Hour time.



```
Enter 24-Hour Time (hh:mm): 13:24
Converted 12-Hour Time: 01:24 PM
```

Code is not just a set of instructions; it's a story that you tell the machine. Embrace the challenges, debug the setbacks, and let your code narrate the tale of your ingenuity.

Keep coding, for in every line, you create a world of possibilities.