Program Persist Data: Group Assignment1

Odd Teams (team numbers 1, 3, 5, 7 & 9 from all lab groups)

General Instructions
Prepare and submit before April 22nd at 5pm.
Equal marks for each task.
This will be demoed in lab and to the class within the teams.

Prepare using C, error check and comment all 'moving parts' appropriately[1]. You are given directions create function names that are appropriate and use functions wherever possible. Reference any code that you use that you have not created or been given in class.
Consider getting a GitHub account and work within a project space for each task.

1. **Read MP3 Information**

You are required to create a 'utility' program called readMP3.c that is called by the command line specifying the name of the file to analyse. For example:

```
C:> read_mp3 mysong.mp3
```

The utility is supposed to get some metadata information from the mp3 file by reading the MP3 audio tag stored in the file, and display the following information on the screen.

```
Title of the song
Author of the song (artist)
Album
Year
Genre
```

Some of the information may not be available to you from the tag. In that case an "Unknown" value should be displayed. Only use MP3 files that contain a valid MP3 audio tag ID3 version 1 (mp3 audio tag ID3v1 format is displayed in the next page).

To achieve the aim you need to read the audio tag at the end of the file, save into the right structure in memory and access the data.
The utility will check if the mp3 file has a valid MP3 audio tag. A valid tag contains the correct audio tag, check it the first 3 bytes contain the letter 'T', 'A', 'G'. If not a valid MP3 file display an error message such as "The file does not contain a valid audio tag" and terminate the program. If the audio tag is valid, you are required to display Title, Artist, Album, Year and Genre.

MPEG Audio Tag ID3v1

The TAG is used to describe the MPEG Audio file. It contains information about artist, title, album, publishing year and genre. There is some extra space for comments. It is exactly 128 bytes long and is located at very end of the audio data. You can get it by reading the last 128 bytes of the MPEG audio file.

AAABBBBB	BBBBBBBB	BBBBBBBB	BBBBBBBB
BCCCCCCC	CCCCCCCC	CCCCCCCC	CCCCCCCD
DDDDDDDD	DDDDDDDD	DDDDDDDD	DDDDDEEE
EFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFG

Sign	Length (bytes)	Position (bytes)	Description
A	3	(0-2)	Tag identification. Must contain 'TAG' if tag exists and is correct.

[1] Over commenting is essential in team exercises.

B	30	(3-32)	Title
C	30	(33-62)	Artist
D	30	(63-92)	Album
E	4	(93-96)	Year
F	30	(97-126)	Comment
G	1	(127)	Genre

The specification asks for all fields to be padded with null character (ASCII 0). However, not all applications respect this (an example is WinAmp which pads fields with <space>, ASCII 32).

There is a small change proposed in **ID3v1.1** structure. The last byte of the Comment field may be used to specify the track number of a song in an album. It should contain a null character (ASCII 0) if the information is unknown.

Genre is a numeric field which may have one of the following values:

0 'Blues'	20 'Alternative'	40 'AlternRock'	60 'Top 40'
1 'Classic Rock'	21 'Ska'	41 'Bass'	61 'Christian Rap'
2 'Country'	22 'Death Metal'	42 'Soul'	62 'Pop/Funk'
3 'Dance'	23 'Pranks'	43 'Punk'	63 'Jungle'
4 'Disco'	24 'Soundtrack'	44 'Space'	64 'Native American'
5 'Funk'	25 'Euro-Techno'	45 'Meditative'	65 'Cabaret'
6 'Grunge'	26 'Ambient'	46 'Instrumental Pop'	66 'New Wave'
7 'Hip-Hop'	27 'Trip-Hop'	47 'Instrumental Rock'	67 'Psychadelic'
8 'Jazz'	28 'Vocal'	48 'Ethnic'	68 'Rave'
9 'Metal'	29 'Jazz+Funk'	49 'Gothic'	69 'Showtunes'
10 'New Age'	30 'Fusion'	50 'Darkwave'	70 'Trailer'
11 'Oldies'	31 'Trance'	51 'Techno-Industrial'	71 'Lo-Fi'
12 'Other'	32 'Classical'	52 'Electronic'	72 'Tribal'
13 'Pop'	33 'Instrumental'	53 'Pop-Folk'	73 'Acid Punk'
14 'R&B'	34 'Acid'	54 'Eurodance'	74 'Acid Jazz'
15 'Rap'	35 'House'	55 'Dream'	75 'Polka'
16 'Reggae'	36 'Game'	56 'Southern Rock'	76 'Retro'
17 'Rock'	37 'Sound Clip'	57 'Comedy'	77 'Musical'
18 'Techno'	38 'Gospel'	58 'Cult'	78 'Rock & Roll'
19 'Industrial'	39 'Noise'	59 'Gangsta'	79 'Hard Rock'

Otherwise “unknown”

2. Inventory implementation

Create a simple IT assets inventory that performs the following functions:

1. Add new item
2. Delete old item
3. Edit item
4. Find item
5. Record inventory

Sample list item:

Quantity	Model	Cost
5	Dell 12345	2390.00

This program is to be implemented using an array. The code should use functions where appropriate and effective error checking. Select an appropriate unique identifier (key).

For the fifth function 'Record inventory' implement an appropriate sorting algorithm based on the design of max 200[2] inventory items and create a file called ITinventory that is used to store the sorted items for reference. An additional variable within the file or filename argument will be required to date the inventory file.

In addition to the required user functions, create an additional display() in order to demonstrate your code after each addition, deletion or alteration. When the inventory is full send an alert to the user that no further items may be added.
In order to demonstrate use an array of 5 with 3 pre-initialised contacts.

3. Queue implementation.

Using a linked list dynamic data structure implement a queue structure IT services ticketing system.
The system will enable the user add tickets when IT fix requests are received via phone.
The important information for the system is as follows:

Name of client:
Contact Number:
Short description of issue:
Time of phone call:

The ticketing system will only allow the next ticket in the queue be processed and the queue has a limit of 5 tickets. Select an appropriate unique identifier for the system[3].

Please initialise 3 tickets in whatever manner you desire in order to demonstrate your system.

Jane	4558	Monitor broken	12.05
Sean	5656	Desktop won't flickering	12.06
Mark	1212	Router lights amber	12.10

Code the system using functions where possible and create an additional display() in order to demonstrate your code after each new input to the queue and each dequeued

[2] Use a symbolic constant so you may alter to demo.
[3] The IT office has 1 phone between departments, their clients have a hot desk environment which means office phone is a shared resource.

(processing of tickets at start of queue). When the queue is full send an alert to the user that no further tickets may be taken.