

Lab 3: Arrays, C-struct, and C-string

Programming for Engineers Lab

Name: Ahmed Ismail | ID: 8541747 | Course Title: CSCI291 | Date: 27/10/2024

Blue text on a black background

Description automatically generated

Table of Contents

[1. Source Code Documentation 2](#_Toc182510164)

[Question 1: Array Utility Functions 2](#_Toc182510165)

[Question 2: Banking Transactions Processing 4](#_Toc182510166)

[Question 3: League Team Application 5](#_Toc182510167)

[2. Design Solutions 7](#_Toc182510168)

[Question 1: Array Utility Functions 7](#_Toc182510169)

[Question 2: Banking Transaction Processing Pseudocode 7](#_Toc182510170)

[Question 2: Banking Transaction Processing Flow chart: 9](#_Toc182510171)

[Question 3: search\_update Function Design 9](#_Toc182510172)

[Question 3: search\_update Function FlowChart 11](#_Toc182510173)

[3. Testing Evidence and Discussion 12](#_Toc182510174)

[Question 1 Testing Results 12](#_Toc182510175)

[Question 2 Testing Results 14](#_Toc182510176)

[Question 3 Testing Results 14](#_Toc182510177)

[GitHub Page: 18](#_Toc182510178)

# 1. Source Code Documentation

## Question 1: Array Utility Functions

**Part 1:**

* isValid: Validates array positions. If pos is valid, return true. Else return false.

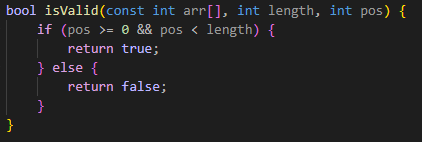


Figure 1 isValid Function code

* remove\_element: Removes elements with upward shifting. Preserves first element as required. If pos is valid, it performs a **leftward shift** starting from the specified position. It uses a for loop to assign each element to the position to its right (arr[i] = arr[i-1]). This "removes" the element at pos by overwriting it.

A computer screen with text and symbols

Description automatically generated

Figure 2 remove\_element Function code

* insert\_element: Inserts elements with downward shifting. If valid, it performs a **rightward shift** to make room for the new value. It uses a for loop to shift each element to the left (arr[i] = arr[i+1]) up to the specified position. This shift causes the first element in the array to be overwritten by the second, and so on, until the new value is inserted at pos. After the loop, the function inserts value into arr[pos].

A computer screen shot of a program code

Description automatically generated

Figure 3 insert\_element Function code

**Part 2:**

* reshape: Converts 1D array to 2D array column-wise. row and col keep track of the current position in the 2D array, with col incrementing until it reaches the number of columns (nCols). When col reaches nCols, it resets to 0, and row increments to move to the next row.

A screen shot of a computer code

Description automatically generated

Figure 4 reshape Function code

* trans\_matrix: Creates transpose of input matrix. The function iterates through each element in the input matrix (mat), switching the row and column indices when assigning values to the transposed matrix (mat\_transp[j][i] = mat[i][j]).

A black screen with white text

Description automatically generated

Figure 5 trans\_matrix Function code

* found\_duplicate: Checks for duplicate values in array. It uses a nested loop to compare each element with all subsequent elements in the array. The outer loop iterates through each element, and the inner loop compares it with all elements that come after it. If a match is found (arr[i] == arr[j]), the function returns true, indicating a duplicate was found.

A computer screen shot of a code

Description automatically generated

Figure 6 found\_duplicate Function code

## Question 2: Banking Transactions Processing

**Code components:**

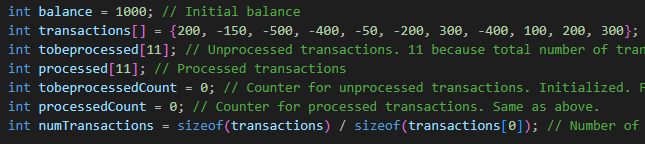
* Begins with initial balance of 1000 AED. Uses parallel arrays for tracking processed/unprocessed transactions. Implements transaction counters.

Figure 7 Initialization of balance, transaction and arrays code

* Implements validation for withdrawals, Handles zero balance condition. Uses separate arrays for tracking transaction status.

A screen shot of a computer program

Description automatically generated

Figure 8 For loop of transaction processing

## Question 3: League Team Application

**Code components:**

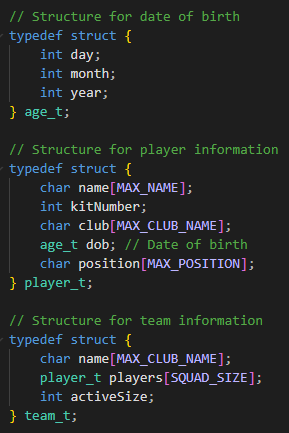
****

Figure 9 Structs declared for Question 3

Nested Structure Design:

* Uses typedef for clean syntax
* Implements hierarchical data organization
* Proper size constraints for strings

Memory Management:

* Fixed-size arrays for predictable memory usage
* No dynamic allocation required
* Efficient structure packing

**Core Functions:**

* enroll\_club:

A computer code on a black background

Description automatically generated

Figure 10 enroll\_club code

* + Maintains team count
  + Validates against maximum teams
  + Uses pointer for count update
* add\_player:
  + Club Selection
  + Play Information Noting
* search\_update:
  + Search Options:
    - Name-based (case-insensitive)
    - Kit number-based
  + Update Options:
    - All player attributes modifiable
    - Maintains data integrity
  + Validation:
    - Duplicate checking
    - Range validation
    - Date validation
* Error Handling:

A black screen with white text

Description automatically generated

Figure 11 Error Handling message maker code

* Validation Functions:
  + Kit Number Validation
  + Date Validation
  + Duplicate Checking

# 2. Design Solutions

## Question 1: Array Utility Functions

Main functionalities:

1. isValid: Checks if a given position in an array is valid based on the array's length.
2. remove\_element: Removes an element from an array by shifting all subsequent elements up.
3. insert\_element: Inserts an element into an array by shifting all subsequent elements down.
4. reshape: Converts a 1D array into a 2D matrix with the specified number of rows and columns.
5. trans\_matrix: Transposes a 2D matrix.
6. found\_duplicate: Checks if an array contains any duplicate elements.

## Question 2: Banking Transaction Processing Pseudocode

Initialize:

Set balance = 1000

Create arrays: transactions[], tobeprocessed[], processed[]

Set counters: tobeprocessedCount = 0, processedCount = 0

Process Transactions:

FOR each transaction in transactions array:

IF transaction is withdrawal AND exceeds balance:

Print invalid transaction message

Add to tobeprocessed array

Continue to next transaction

IF balance is 0:

Print zero balance message

Add remaining transactions to tobeprocessed

Break loop

Update balance with transaction

Add to processed array

Increment processed counter

Output Results:

Print final balance

Print all processed transactions

Print all unprocessed transactions

## Question 2: Banking Transaction Processing Flow chart:

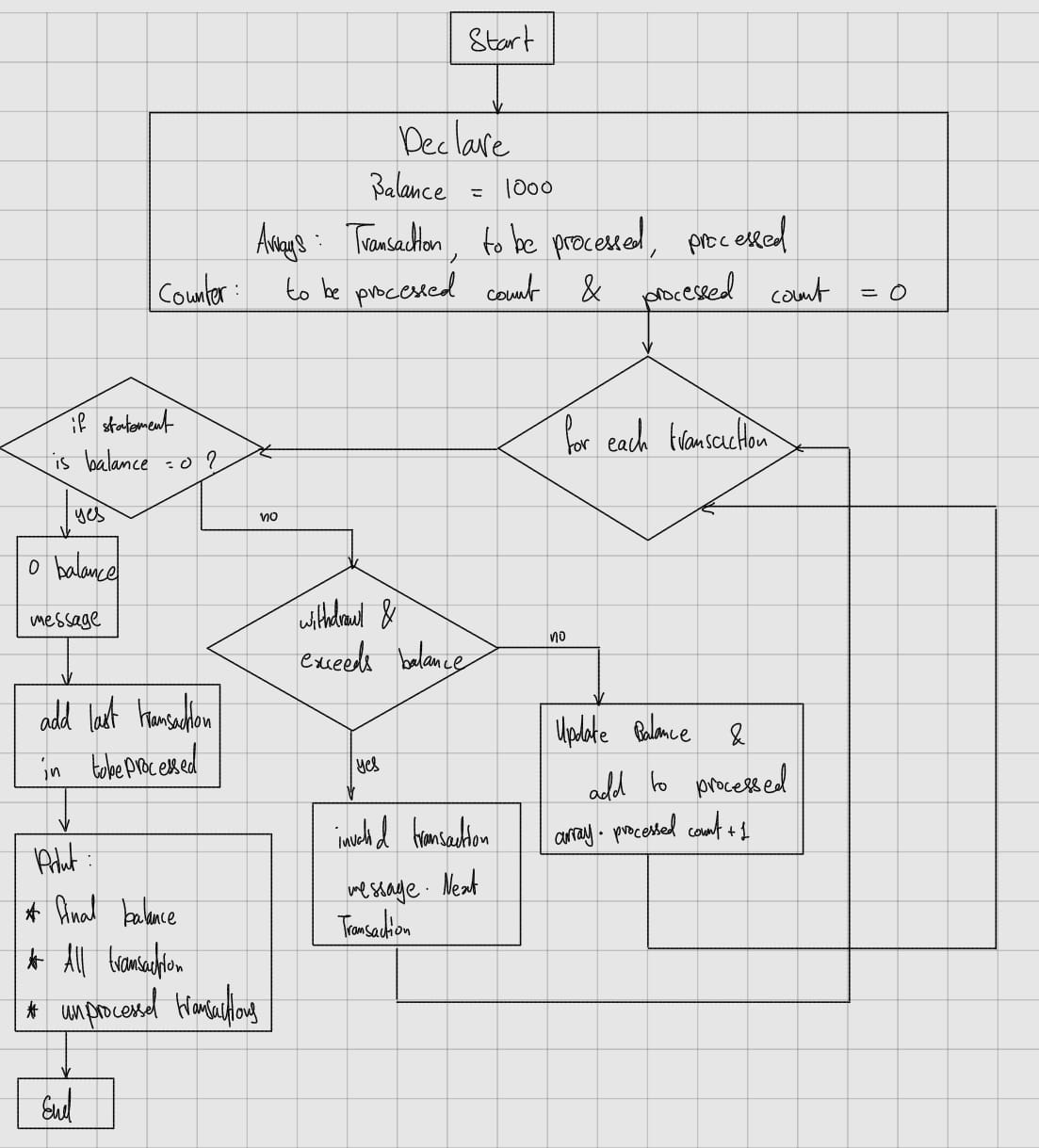


Figure 12 Question 2 Code Flow Chart

## Question 3: search\_update Function Design

Structure Access Explanation:

1. The function accesses struct data through nested structures:

team\_t teams[] -> player\_t players[] -> Individual fields

Data access patterns:

* Direct access: teams[i].name
* Nested access: teams[i].players[j].name
* Pointer access: player\_t \*player = &teams[foundTeamIndex].players[foundPlayerIndex]

The search\_update function follows this workflow:

1. Takes user input for search method (name/kit number)
2. Traverses team and player arrays using nested loops
3. Uses pointer to found player for efficient updates
4. Implements validation before updates
5. Maintains data integrity through duplicate checks

## Question 3: search\_update Function FlowChart

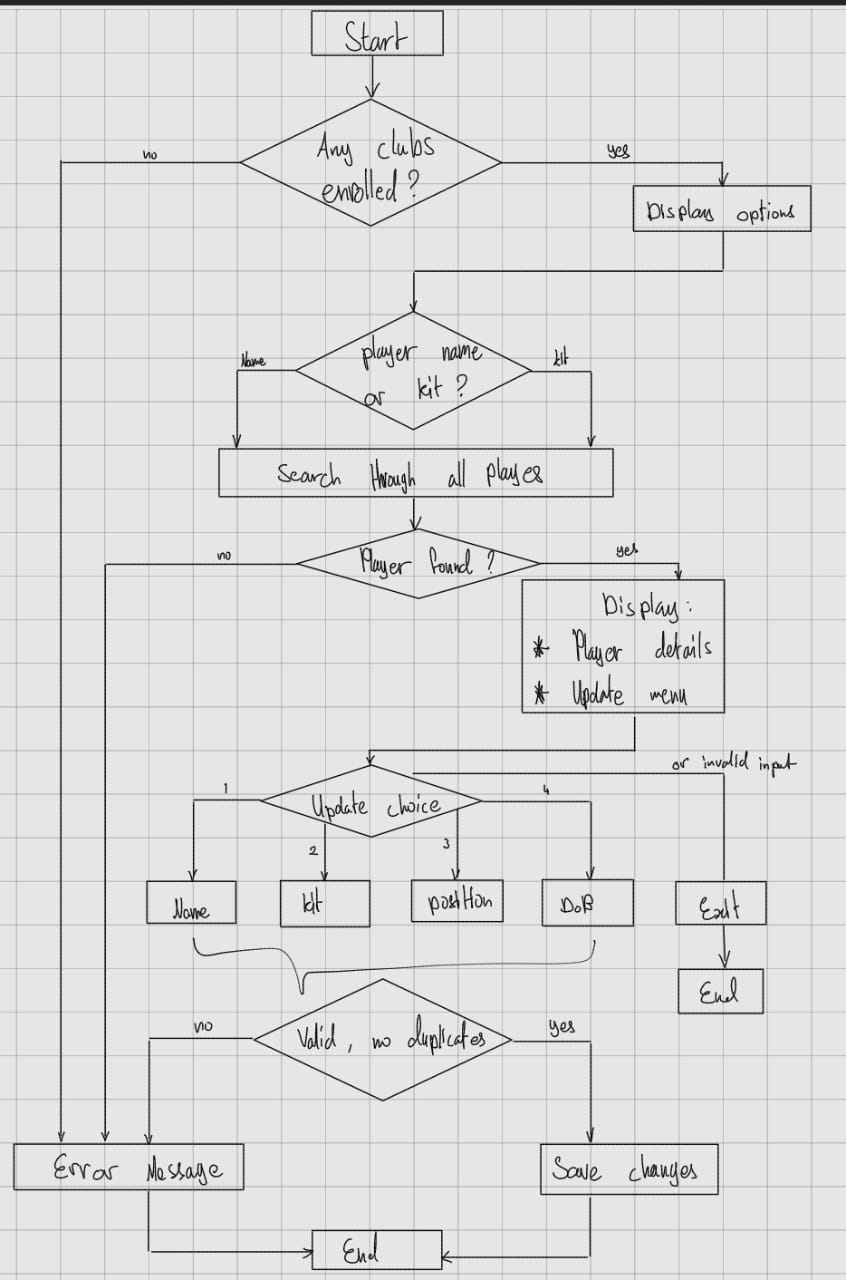


Figure 13 Question 3 Code Flow Chart

# 3. Testing Evidence and Discussion

## Question 1 Testing Results

Test case 1: isValid function testing

A black screen with white text

Description automatically generated

Figure 14 isValid Test Evidence

Test Case 2: remove\_element demonstration

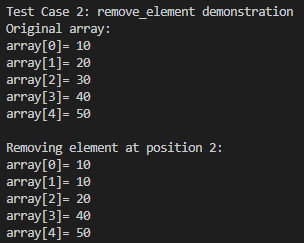


Figure 15 remove\_element Test Evidence

Test Case 3: remove\_element with invalid position

A black background with white text

Description automatically generated

Figure 16 remove\_element invalid Test Evidence

Test Case 4: insert\_element demonstration

A computer screen shot of a black screen

Description automatically generated

Figure 17 insert\_element Test Evidence

Test Case 5: insert\_element with invalid position

A black background with white text

Description automatically generated

Figure 18 insert\_element invalid Test Evidence

Test case 6: Reshaped 2D array

A black screen with white text

Description automatically generated

Figure 19 reshape Test Evidence

Test case 7: Transposed matrix

A black screen with white text

Description automatically generated

Figure 20 trans\_matrix Test Evidence

Test case 8: found\_duplicate



Figure 21 found\_duplicate Test Evidence

## Question 2 Testing Results

A screenshot of a computer

Description automatically generated

Figure 22 Question 2 Test Evidence

## Question 3 Testing Results

Club enrolment:

A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated

Figure 23 Enrolment Test Evidence

Player addition:

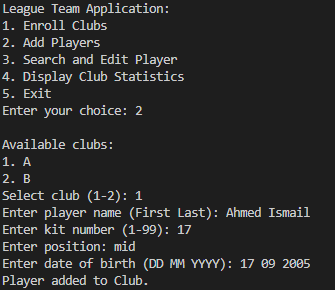


Figure 24 Player addition Test Evidence

A screen shot of a computer

Description automatically generated

Figure 25 Player addition proof in Club Statistics

Search and update functionality verified:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Figure 26 Update Functionality for several cases Test Evidences

A screenshot of a computer

Description automatically generatedA black background with white text

Description automatically generatedA screenshot of a computer program

Description automatically generated

Figure 27 More Update Functionality Test Evidences

Statistics display correctly formatted:

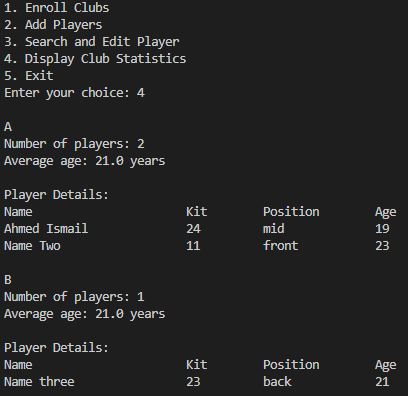


Figure 28 Statistics Display Test Evidence

Invalid Entries Tests:

A screen shot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

Figure 29 Invalid Entries, Duplicates and such Test Evidence

# GitHub Page:

*https://github.com/AIsmail17/CSCI291-Lab-3-Evidence.git*