# CSCI291, Task 2: Arrays, C-struct, and C-string (100 marks) Due date: 15/11/2024 @ 11:55 pm

**Constraint: C pointer variables must NOT be used to solve any of the questions in this assessment.**

# Question 1: Array Utility Functions [19 marks]

Consider the attached C program Lab3\_Q1.c where *arr* is a 1D integer array of length SIZE, and *arr2d* is a 2D integer array of *nRows* rows and *nCols* columns.

After coding in **C** the below functions, complete the main() function to test the correctness of the function implementations. Your program should not use a global variable or additional define symbolic constant. Note that you **might** need to add further parameters to the below function signatures/headers

* **bool isValid(const int arr[], int length, int pos,…**): returns true if pos is within the valid range of array indices, false otherwise. **This function must be called in every function that checks the validity of the indices.**

## [2 marks]

* **void remove\_element(int arr[], int length, int pos):** removes the array element at index *pos* by shifting up all preceding array elements by one position, see Fig 1.1. The value of the array at index 0 remains unchanged. If “pos” is not a valid array index, print a relevant message and exit the function.

|  |  |  |
| --- | --- | --- |
| 10 | | |
| 20 | | |
|  | 30 |  |
|  | 40 |  |
|  | 50 |  |

|  |  |  |
| --- | --- | --- |
| 10 | | |
|  | 10 |  |
|  | 20 |  |
| 40 | | |
| 50 | | |

## Fig 1.1: remove\_element of arr[2]

**[4 marks]**

* **void insert\_element(int arr[],int length, int pos, int value):** inserts the parameter value at the specified index *pos*, shifting the original value at pos and all its preceding elements one position down, see Fig 1.2. If the value of *pos* is not a valid array index, print a relevant message and exit the function.

|  |  |  |
| --- | --- | --- |
| 10 | | |
|  | 20 |  |
|  | 30 |  |
| 40 | | |
| 50 | | |

|  |  |  |
| --- | --- | --- |
|  | 20 |  |
|  | 30 |  |
| **80** | | |
| 40 | | |
| 50 | | |

**Fig 1.2:** Insert the value 80 at array index 2

## [3 marks]

* **void reshape(const int arr[], int length, int nRows, int nCols, int arr2d[nRows][nCols**])

implements the two requirements:

* + If the length of *arr* is not equal “nRows\*nCols”, print a relevant error message and exit the

function.

* + Otherwise, copy the elements of the 1-D array *arr* into *arr2d*, filling it **column by column.**

## [4 marks]

* **void trans\_matrix(int nRows, int nCols, const int mat[nRows][nCols], int mat[nCols][ nRows]):** generates mat\_transp, the transpose of the input matrix mat.

## [2.5 marks]

* **bool found\_duplicate(int arr[], int length, ….):** returns *true* if there is at least a duplicate values in *arr*; otherwise returns *false*.

**[3.5 marks]**