LAB 02: Arrays

CS211 – Data Structures and Algorithms
Usman Institute of Technology
Fall 2021

- **How to submit:**
  - Submit lab work in a **single** .py file on MS Teams. (No other format will be accepted)
  - Lab work file name should be saved with your roll number (e.g. 19b-001-SE_LW.py)
  - Submit home work in a **single** .py file on MS Teams. (No other format will be accepted)
  - Lab work file name should be saved with your roll number (e.g. 19b-001-SE_HW.py)

1. **Create a parent class Array which takes one parameter to initialize: of an array. Use `ctypes` library to initialize size of array.**

   a) Add a constructor of the class that initializes a size of an array. All elements must be declared **None** by default.
   b) Define a function `__len(self)__` that returns size of an array.
   c) Define a function `__getitem__(self,index)` that gets the contents of the index element
   d) Define a function `__setitem__(self,index,value)` that puts the value in the array element at index position.
   e) Define a function `clear(self,value)` that Clears the array by setting each element to the given value.
   f) Define a function `__iter__(self)` that Returns the array's iterator for traversing the elements.
   g) Declare a class _Array`Iterator` that iterate over the elements of an array.

2. **Create a parent class <u>Array2D</u> which takes two parameters to initialize: <u>rows and columns</u> and write functions in Python whose parameters and return value are given below.**

a) Add a constructor of the class that initializes a list containing $rows \times cols$ elements. All elements must be declared 0 by default.
b) Define a function `__numRows(self)__` that returns number of rows in 2D-array.
c) Define a function `__numCols(self)__` that returns number of columns in 2D-array.
d) Define a function `__getitem__(self,ndxTuple)` that gets the contents of the index element.
e) Define a function `__setitem__(self,ndxTuple,value)` that puts the value in the array element at index position.

f) Define a function `clear(self,value)` that Clears the array by setting each element to the given value.

3. Implement Matrix class that can perform following operations,

A matrix is a collection of scalar values arranged in rows and columns as a rectangular grid of a fixed size. The elements of the matrix can be accessed by specifying a given row and column index with indices starting at 0.

- `Matrix( rows, ncols )`: Creates a new matrix containing nrows and ncols with each element initialized to 0.
- `numRows()`: Returns the number of rows in the matrix.
- numCols(): Returns the number of columns in the matrix.
- `getitem ( row, col )`: Returns the value stored in the given matrix element. Both row and col must be within the valid range.
- `setitem ( row, col, scalar )`: Sets the matrix element at the given row and col to scalar. The element indices must be within the valid range.
- `scaleBy( scalar )`: Multiplies each element of the matrix by the given scalar value. The matrix is modified by this operation.
- `transpose()`: Returns a new matrix that is the transpose of this matrix.
- `add ( rhsMatrix )`: Creates and returns a new matrix that is the result of adding this matrix to the given rhsMatrix. The size of the two matrices must be the same.
- `subtract ( rhsMatrix )`: The same as the add() operation but subtracts the two matrices.
- `multiply ( rhsMatrix )`: Creates and returns a new matrix that is the result of multiplying this matrix to the given rhsMatrix. The two matrices must be of appropriate sizes as defined for matrix multiplication.

Home Task

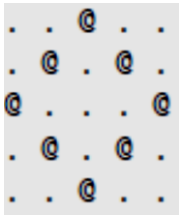
1. A grayscale digital image is a two-dimensional raster image in which the picture elements, or pixels, store a single value representing a shade of gray that varies from black to white. In a discrete grayscale image, the shades of gray are represented by integer values in the range [0 … 255], where 0 is black and 255 is white. We can define the Grayscale Image ADT for storing and manipulating discrete grayscale digital images. Given the description of the operations, provide a complete implementation of the ADT using a 2-D array.
    - `GrayscaleImage( nrows, ncols )`: Creates a new instance that consists of nrows and ncols of pixels each set to an initial value of 0.
    - `width()`: Returns the width of the image.
    - `height()`: Returns the height of the image.
    - `clear( value )`: Clears the entire image by setting each pixel to the given intensity value. The intensity value will be clamped to 0 or 255 if it is less than 0 or greater than 255, respectively.
    - `getitem ( row, col )`: Returns the intensity level of the given pixel. The pixel coordinates must be within the valid range.

- `setitem ( row, col, value )`: Sets the intensity level of the given pixel to the given value. The pixel coordinates must be within the valid range. The intensity value is clamped to 0 or 255 if it is outside the valid range.

2. Implement Matrix class that can perform following operations,
   - Addition
   - Subtraction
   - Scaling
   - Transpose
   - Multiplication
   - Inverse of Matrix

3. Implement the numLiveNeighbors() method of the LifeGrid class.

4. Complete the implementation of the gameoflife.py program by implementing the draw() function. The output should look similar to the following, where dead cells are indicated using a period and live cells are indicated using the @ symbol.



5. Modify the gameoflife.py program to prompt the user for the grid size and the number of generations to evolve.