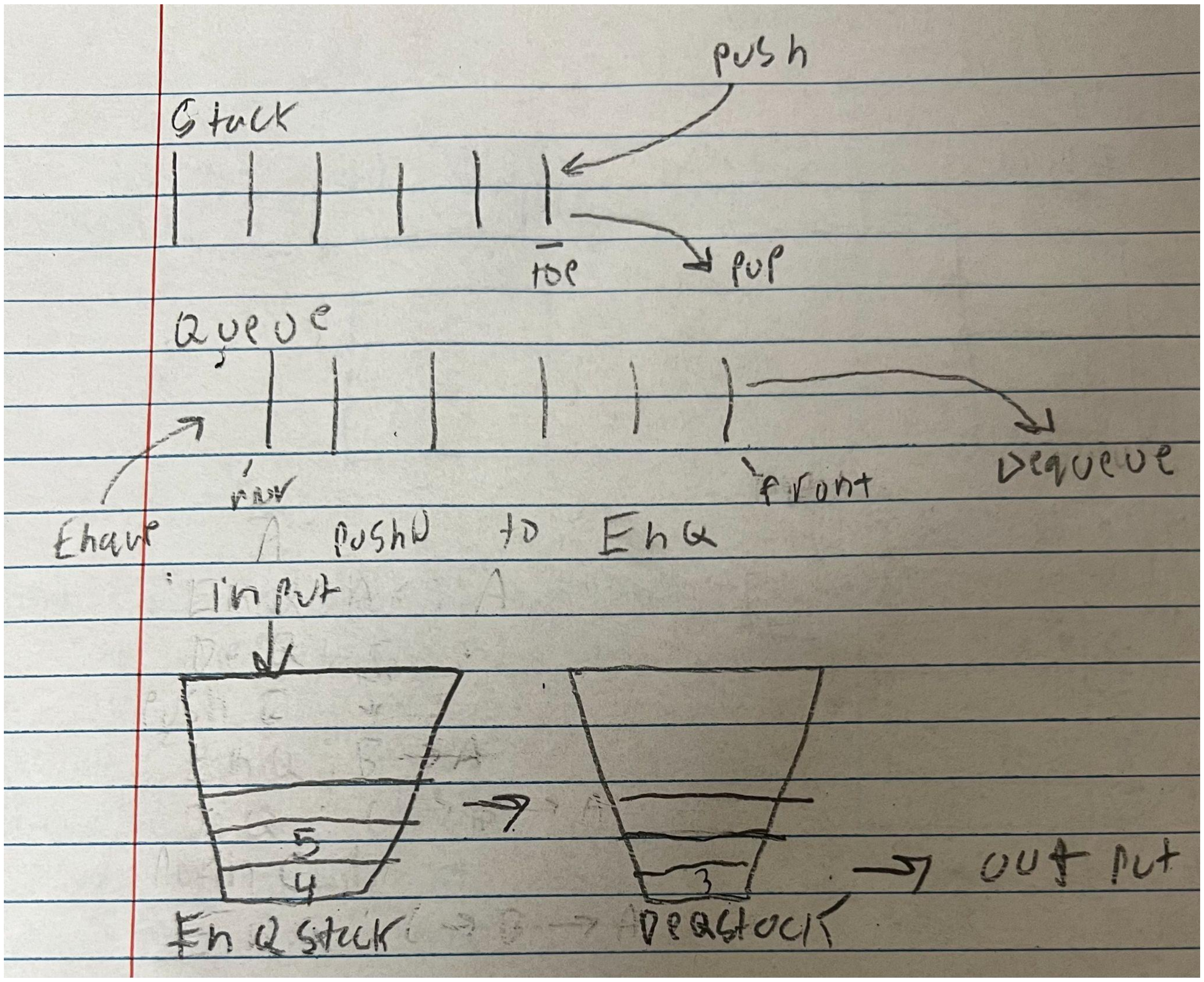
Author Name: Eli Mishriki
Project Structure:

- The Data class:
  - Variables: String fileName, float fileSize
  - Constructor: declares fileName, fileSize
  - Print: prints the fileName and fileSize
- The Node:
  - Variables: T *data, Node<T> *nextNode
  - Constructor: declares data, sets nextNode to nullptr
  - Print: print from data class
- The Stack Container LLStack:
  - Private:
    - Variables: Node<T> *top, int stackSize, const int SMAXITEMS =100
  - Public:
    - Constructor: sets top to nullptr, stackSize to 0
    - Methods:
      - bool isFull(): returns true if stackSize is equal to SMAXITEMS otherwise false
      - bool isEmpty(): returns true if the top node is empty. false otherwise
      - void push(T* item): accepts a pointer to an item of an object type T returns void. If the stack is not full create a node called newNode of type T that contains Item the new nodes pointer is set to the top of the stack. The top of the stack is then updated to be this new node. StackSize is incremented by 1. If full returns “Stack Overflow”
      - void pop(): takes no argument if the stack is not empty and sets a temp node to the top of the stack. Sets the top pointer to the nextNode in the stack. Deletes temp. Decrementes stack size by 1. Returns “Stack Underflow” if the stack is empty.
      - T* peek(): takes no argument if the stack is not empty and returns the data in the top node otherwise returns “Stack Empty”
- The Implementation StackQ
  - Private:
    - Variables: LLStack<T>* enQStack, LLStack<T>* deQStack, int queueSize, const int QMAXITEMS
  - Public:
    - Constructor: initializes an instance of the StackQ class creating two new stacks enQStack and deQStack of type LLStack<T>. Sets queue size to 0.
    - Deconstructor: Deletes the two stacks made in the constructor.
    - Methods
      - bool isFull(): returns true if stackSize is equal to QMAXITEMS otherwise false

        - bool isEmpty(): returns true if the queue is empty
        - void enqueue(T* item): Takes in an item of type T if the stack is not full pushing the item to the top of enQStack and increments the queue by 1. If the queue is full returns Queue overflow
        - void dequeue(): takes no argument if the stack is not full does the following. If deQstack is empty, we push all items from enQStack to deQstack while popping them from enQStack. Then it deletes deQstack peek and pops the top item. queusize is decremented by 1. If the stack is empty returns “Queue Underflow”
        - T* peek(): checks if the queue is empty if not does the following: if deQStack is empty it transfers all elements from enQStack to deQStack so the top element of deQStack is the front of the queue. Then peeks deQStack and returns it since this element is the front of the queue
        - int getQueueSize(): returns the current size of the queue
        - void displayQueue(): checks if empty if not does the following: if deQStack is empty then it means the latest elements are in enQStack. Transfers all elements from enQStack to deQStack. Transfers items from deQStack to a temporary stack. Prints every element of temp pushes elements back to deQStack and pops elements from temp;
        - void displayStacks: called displayStackItems for both enQStack and deQStack
        - void displayStackItems(LLStack<T>* stack): checks if the stack is empty if not does the following: transfers items from a given stack from argument to a temporary stack prints the items and then transfers them back to the original stack.
- Main Method:
    - creates an instance of StackQ with the template PrintItem
    - displays the menu options a - g
    - asks for user input
        - a: asks for file name and size calls printQueue’s enqueue with filename and file size
        - b: calls printQueue’s deque
        - c: checks if the peek is null otherwise shows peek
        - d: calls prinqueu display queu
        - e: displays print queue size
        - f: calls print queue display stacks
        - g: exits
        - default: invalid choice

Project Definition:


push
Stack
top
pop
Queue
Enque
rear
front
Dequeue
pushed to EnQ
Input
5
4
3
EnQ stack
DeQstack
output