Name: _____ NetID: _____

- **WRITE your name and NetID on EVERY page.**
- **DO NOT REMOVE** THE STAPLE IN YOUR EXAM.
- **DO NOT BEGIN** UNTIL INSTRUCTED TO DO SO.
- WRITE NEATLY AND CLEARLY. If we cannot read your handwriting, you will not receive credit. Please plan your space usage. No additional paper will be given.
- This exam is worth 150 points.

Problem 1 – Arrays (15 points)

The Game of Life assignment consisted of a board (grid) of ***rows*** *x* ***cols*** cells, each in one of two states, *alive* or *dead*.
Answer the questions below about the following constructor implementation.

```
public GameOfLife (String file) {

    StdIn.setFile(file);
    int row = StdIn.readInt();
    int col = StdIn.readInt();

    grid = new boolean[row][col];
    totalAliveCells = 0;

    for ( int curRow = 0; curRow < row; curRow++ ) {

        for ( int curCol = 0; curCol < col; curCol++ ) {

            grid[curRow][curCol] = StdIn.readBoolean();

            if ( grid[curRow][curCol] == ALIVE ) {
                totalAliveCells += 1;
            }
        }
    }
}
```

a) **(5 points)** What is the total number of array accesses (reads and writes to the grid array) as a f unction of *rows* and *cols*?
f(rows,cols) = 2 * rows * cols // 1 read and 1 write per iteration of the inner loop.
OR
f(rows,cols = 4 * rows * cols // 3 reads (2 read column reference to access column, 1 read column) and 1 write column per iteration of the inner loop.

b) **(5 points)** What is the tilde notation of the function from part (a)?
~2 * rows * cols OR ~4 * rows * cols
// correct if this part has the correct tilde notation of part a. Even if part a is incorrect

c) **(5 points)** What is the big O notation of the function from part (a)? O (rows * cols)
// correct if this part has the correct big O notation of part a. Even if part a is incorrect

Name: _____ NetID: _____

Problem 2 – Search (15 points)

Suppose that you have an unsorted array A of size *n* and a sorted array B of size *m*. Note that *n* is much larger than *m*. Answer the questions below.

(a) **(10 points)** What is the fastest algorithm to find the common values in these two arrays? Describe the algorithm. **No code**, just a succinct description.

Sol: for each value in A do a binary search on B

(b) **(5 points)** What is the worst case big O running time for your algorithm?

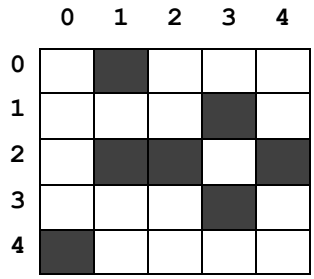
Sol: O(n log m)

Problem 3 – Union-find (30 points)

The Game of Life assignment consisted of a board (grid) of ***n*** *x* ***m*** cells, each in one of two states, *alive* or *dead*. Recall that the board wraps around horizontally and vertically. For two cells to be neighbors they touch vertically, horizontally, or diagonally.

Assume that two neighboring alive cells *a* and *b* have been connected using the `union(aRow, aCol, bRow, bCol)`. The find method signature is: `int find(int row, int col)`

Answer the questions below about the following grid where dark cells are alive and white cells are dead.



(a) (4 points) Does `find(0,1)` and `find(0,3)` return the same values (Yes or No)? NO
(b) (4 points) Does `find(0,1)` and `find(4,0)` return the same values (Yes or No)? YES
(c) (4 points) Does `find(1,3)` and `find(3,3)` return the same values (Yes or No)? YES
(d) (4 points) Does `find(2,4)` and `find(4,0)` return the same values (Yes or No)? NO
(e) (4 points) Does `find(2,3)` and `find(1,4)` return the same values (Yes or No)? NO
(f) (4 points) Does `find(3,3)` and `find(2,4)` return the same values (Yes or No)? YES
(g) (6 points) What is the number of communities (connected components) in the grid? 2

Name: _____ NetID: _____

Problem 4 – Stack (30 points)

(a) (10 points) The following code is supposed to implement a stack (constructor and the push operation) using resizing array but there is an error. Find the error and fix it.

```
1  public ResizingArrayStackOfStrings () {
2      s = new String[1];
3  }
4
5  public void push ( String item ) {
6     if ( n == s.length )
7         resize(2 * s.length);
8
9     s[n++] = item;
10 }
11
12 private void resize ( int capacity ){
13    String[] copy = new String[capacity];
14
15    for ( int i = 0; i < n; i++ )
16        copy[i] = s[i];
17 }
```

Sol: Insert s = copy; after line 16.

(b) (10 points) What is the worst case running time for the push operation using big O notation in (a)? Give the reasoning.

Sol: O(n) - 5 pts

When trying to insert a item into a full stack, we need to create a new array with double size and copy over all the elements from the old array into the new array. - 5 pts

(c) (10 points) A stack can be used to check whether a given expression has balanced parentheses or not. An expression is parsed from left to right. When parsing, if you see an open parenthesis, push it into the stack. If you see a close parenthesis, pop the top element (if exists) from the stack. After you have parsed all characters in the expression, check the stack. An empty stack means that the parenthesis match, a non-empty stack means parenthesis do not match. Which of the following will result in the non-empty stack.

i. (1+2)*(3+4)
ii. 15/5*(3+4)*6)
iii. (4*3+(1+2))*5
iv. (5+3*5)+(3*6
// consider correct if students mark ii and iv

Name: _____ NetID: _____

Problem 5 – Special Linked Lists (35 points)

(a) **(5 points)** What is the running time (big O notation) to insert a new node in the front of a Circular Linked List (assume a reference to the end of the list)? Give the reasoning.

[2 points] O(1)
[3 points] To insert at the front requires creating a new node and updating the last pointer to the new front. No list traversal is needed.

(b) **(5 points)** What is the worst case running time (big O notation) to insert a new node in a sorted Doubly Linked List? Give the reasoning.

[2 points] O(n)
[3 points] Item to be inserted is greater than all other items in the list. Need to traverse the entire list to insert item.

(c) **(25 points)** On the RUKindergarten assignment there are two linked lists of `SNode`:

- `studentsInLine`: a singly linked list
- `musicalChairs`: a circular linked list

```
public class SNode {
    public Student student; // the data part of the node
    public SNode   next;    // a link to the next student int the linked list

    public SNode ( Student s, SNode n ) {
                student = s;
                next = n;
        }
}
```

For this question assume that `Classroom` contains the following methods.

```
public class Classroom {
    private SNode studentsInLine;  // reference to the FIRST student in the LL
    private SNode musicalChairs;   // reference to the LAST student in the CLL


    // Removes and returns the first student in studentsInLine
    public Student removeFirstStudentFromLine () {}

    // inserts the argument Student as the last student in musicalChairs
    public void insertStudentIntoLastChair (Student s) {}

    public void moveStudentsFromLineIntoChairs () {

        while ( studentsInLine != null ) {

            Student s = removeFirstStudentFromLine();
            insertStudentIntoLastChair(s);
        }
    }
```

Name: _____ NetID: _____

(10 points) Implement the method `removeFirstStudentFromLine`. The method can be used in other methods besides `moveStudentsFromLineIntoChairs`.

```
public Student removeFirstStudentFromLine () {
   // COMPLETE THIS METHOD
```

```
// Removes and returns the first student in studentsInLine
public Student removeFirstStudentFromLine () {

    if ( studentsInLine == null ) return null; // 3 points

    Student s = studentsInLine.student;
    studentsInLine = studentsInLine.next; // 5 points: removes front node and updates front
    returns s; // 2 points: returns the removed students

}
```

(15 points) Implement the method `insertStudentsIntoLastChair`, the method can be used in other methods besides `moveStudentsFromLineIntoChairs`.

```
public void insertStudentsIntoLastChair ( Student s )  {
   // COMPLETE THIS METHOD
```

```
// inserts the argument Student as the last student in musicalChairs
public void insertStudentIntoLastChair (Student s) {

    SNode oldLast       = musicalChairs;
    SNode musicalChairs = SNode (s, null); // 2 points: creates new node

    if ( oldLast == null ) {
        // list is empty
        musicalChairs.next = musicalChairs;// 5 points: makes only node point to itself on empty list
    } else {
        musicalChairs.next = oldLast.next: // 5 points: new last points to front
        oldLast.next  =  musicalChairs;   // 3 points: previously last node is next to last
    }
}
```

Name: _____ NetID: _____

Problem 6 – Linked Lists (25 points)

Implement the following method to delete the last occurrence of an item from a linked list. For example:

Input: `23 -> 12 -> 23 -> 6`

Delete last occurrence of 23, resulting in the list: `23-> 12 -> 6`

You may NOT use or implement helper methods - all your code must be implemented inside the given method. You may NOT use recursion.

```
public class Node {
    public int data;
    public Node next;
}

// Deletes LAST occurrence of given item from a linked list,
// given a front pointer to the first node in the list.
// Returns pointer to first node of the updated linked list.
// Input list could be empty.
// Throws NoSuchElementException if item is not in the linked list.

public static Node deleteLastOccurrence (Node front, int item)
throws NoSuchElementException {

    // COMPLETE THIS METHOD

}
```

```
public static Node deleteLastOccurrence(Node front, int item)

     throws NoSuchElementException {

        // COMPLETE THIS METHOD
        Node ptr=front, prev=null, match=null, matchprev = null;

        // find last occurrence
        while (ptr != null) {

           if (ptr.data == item) {
              match = ptr;
              matchprev = prev;
           }

           prev = ptr;

           ptr = ptr.next;

        }

        if (match == null) {  //no match (includes empty LL case)
           throw new NoSuchElementException();
        }

          if (matchprev == null) { //only one occurrence, at first node
                return match.next;
          }
```

Name: _____ NetID: _____

```
            matchprev.next = match.next;

            return front;
        }
```

- 4 pts: Proper declaration and initialization of all variables
- 8 pts: Logic for match happening at first node
- 8 pts: Logic for match happening elsewhere
- 5 pts: Returning pointer first node of updated list