EECE.2160 Fall 2018: Exam 3
Structure Definitions and Function Test Cases

Question 1a (`Elem` structure definition; `setElem()` function definition)

```
typedef struct {
     char c;
     int n;
} Elem;

void setElem(Elem *p, char ch, int v) {
     p->c = ch;
     p->n = v;
}
```

Question 1b & 1c (`Point` and `Quadrilateral` structure definitions)

```
typedef struct {
   double x;    // X coordinate
   double y;    // Y coordinate
} Point;

typedef struct {
   Point vert[4];
} Quadrilateral;
```

The `Quadrilateral` structure is a more general version of `Rectangle` from Program 8—it's a collection of four vertices, but you can't assume those four vertices represent a rectangle. Opposite sides may have different lengths, and the sides do not have to be strictly horizontal and vertical.

All you can assume is the vertices are ordered such that the four sides of the quadrilateral are between `vert[0]` and `vert[1]`, `vert[1]` and `vert[2]`, `vert[2]` and `vert[3]`, and `vert[3]` and `vert[0]`.

SEE THE OTHER SIDE OF THIS SHEET FOR INFORMATION RELATED TO QUESTIONS 2a & 5.

Question 2a (`maxFirst()` test cases)

In all cases, assume the function is called as follows:

```
    maxFirst("test_in.txt", "test_out.txt");
```

Input file contents **(`test_in.txt`)**	**Output file contents** **(`test_out.txt`)**	**Return value**

`44 16 -1 1`	`44 16`
	`1 -1`
		4

`12 17 11 41 9 8`	`17 12`
	`41 11`
	`9 8`
		6

`17 20 35 19 99 100 -3 -1 21 60`	`20 17`
	`35 19`
	`100 99`
	`-1 -3`
	`60 21`
		10

Question 5 (Extra Credit): File Contents and Additional Hints

The table below shows the contents of the three files used in this problem:

Name	**Contents**
`FileA.txt`	`Yrx0w4rX2or`
`FileB.txt`	`ouns8sXr42t!`
`FileC.txt`	`A1B2A3B2A4B3A2B1A1B4`

Additional hints:

- The character `'0'` (the digit 0, not the null character) has the ASCII value 48.
- All digits have consecutive ASCII values, so `'1'` = 49, `'2'` = 50, and so on.
- All letters of the same case have consecutive ASCII values
  - So, since `'a'` = 97, we know `'b'` = 98, `'c'` = 99, and so on.
- The program uses two functions from the `<ctype.h>` library we discussed. Assuming the variable `c` is a character:
  - `isdigit(c)` returns true if `c` is a digit (any character from `'0'` to `'9'`)
  - `tolower(c)` operates as follows:
    - If `c` is an uppercase letter, `tolower(c)` returns the lowercase version. For example, `tolower('A') = 'a'`.
    - If `c` is any other character, `tolower(c)` just returns that character. For example, `tolower('b')` returns `'b'`, and `tolower(' ')` returns `' '`.