EECE.3220 Spring 2019: Exam 2
Class Definitions and ADT Descriptions

Class definitions for Section 1

```
class Qscore {        // Class to represent a single question score
public:
   Qscore(int p, int m);              // Constructor
   void display(int i, ostream &out); // Print question #i to out
   double fraction();                 // Fraction of points earned
   int getMax();                      // Access data member max
private:
   int pts;      // Points earned
   int max;      // Max score on question
};

class Tscore {        // Class to represent a test score
public:
   Tscore(int q1s, int q1m, int q2s,        // Constructor
          int q2m, int q3s, int q3m);
   void display(ostream &out);              // Print data to out
private:
   Qscore Q1, Q2, Q3;      // Scores for 3 questions
};
```

Selected function definitions:

```
Qscore::Qscore(int p, int m) : pts(p), max(m) {}

double Qscore::fraction() {
   return (double)pts / max;     // (double) converts pts to double
}                                //   so result doesn't truncate

int Qscore::getMax() {
   return max;
}
```

Test cases:

Code snippet	*Output from display() function*

`Tscore T1(5, 10, 20, 30, 30, 40);`
`T1.display(cout);`
	`Q1: 5/10 (50%)`
	`Q2: 20/30 (66.6667%)`
	`Q3: 30/40 (75%)`
	`TOTAL: 55/80 (68.75%)`

`Tscore T2(25, 30, 19, 20, 46, 50);`
`T2.display(cout);`
	`Q1: 25/30 (83.3333%)`
	`Q2: 19/20 (95%)`
	`Q3: 46/50 (92%)`
	`TOTAL: 90/100 (90%)`

Class definition for Section 3

The `Stack` class definition below is the array-based stack we developed in class. Function definitions aren't shown because I assume you understand what each stack function does.

```
class Stack {
public:
   Stack(unsigned maxSize = 1024);     // Constructor
   ~Stack();                           // Destructor
   bool empty() const;                 // Returns true if stack empty
   void push(const double &val);       // Push val to top of stack
   void pop();                         // Remove top of stack
   double top();                       // Read contents of top of stack
private:
   double *list;      // The actual data stored on the stack
   int tos;           // Index for top of stack
   unsigned cap;      // Capacity (max size) of stack
};
```