EECE.2160 Fall 2017: Exam 3
Structure Definitions and Function Test Cases

Question 1a (`Struct1` and `Struct2` structure definitions)

```
typedef struct {
   int v1;
   int v2;
} Struct1;

typedef struct {
   int v3;
   Struct1 s1;
} Struct2;
```

Question 1b (`GradeList` structure definition, test cases for `semesterGPA()`)

```
typedef struct {
   unsigned int nClass;      // # classes student is taking
                             // 1 <= nClass <= 10
   unsigned int nCred[10];   // Number of credits per class
   char grade[10];           // Grade in each class
} GradeList;
```

`semesterGPA()` test cases:

Contents of structure `glp` points to	Calculation of function return value

`nClass = 3`
`nCred = {2, 3, 4}`
`grade = {'A', 'B', 'C'}`
	`Overall GPA`
	`=(2*4 + 3*3 + 4*2) / (2 + 3 + 4)`
	`=(8 + 9 + 8) / 9 = 2.78`

`nClass = 4`
`nCred = {1, 3, 4, 2}`
`grade = {'A', 'D',`
`'A', 'B'}`
	`Overall GPA`
	`=(1*4 + 3*1 + 4*4 + 2*3) / (1 + 3 + 4 + 2)`
	`=(4 + 3 + 16 + 6) / 10 = 2.9`

SEE OTHER SIDE FOR TEST CASES FOR QUESTION 2B

EECE.2160 Fall 2017: Exam 3
Structure Definitions and Function Test Cases

Question 2b (test cases for `TicTacToe()`)

Each input file holds nine rows, and each row contains one letter and a pair of integers:

- The character is either the letter 'X' or the letter 'O'
- The integers represent a row and column within the 3x3 tic-tac-toe "board"

Input file contents	Output file contents
X 0 0 X 2 0 X 1 0 O 1 1 O 2 1 X 0 1 O 0 2 X 2 2 O 1 2	X X O X O O X O X
X 2 1 O 0 0 X 1 1 O 0 1 X 0 2 O 2 0 X 1 0 O 1 2 X 2 2	O O X X X O O X X

SEE NEXT PAGE FOR STRUCTURE/FUNCTION DEFINITIONS FOR QUESTION 5

EECE.2160 Fall 2017: Exam 3
Extra Credit Problem: Structure and Function Definitions

Question 5 (`Node` structure, `add1` and `add2` functions)

```
typedef struct n {
   int val;
   char ch;
   struct n *next;
} Node;


Node *add1(Node *list, Node *n) {
   n->next = list;
   return n;
}


Node *add2(Node *list, Node *n) {
   Node *prev = NULL;
   Node *cur = list;

   while (cur != NULL && cur->val < n->val) {
      prev = cur;
      cur = cur->next;
   }

   n->next = cur;

   if (prev == NULL)
      list = n;
   else
      prev->next = n;

   return list;
}
```