ENEE 140, Spring 2014
Midterm Exam — Answer Key

Do Not Make a Copy!!

1 (12):	
2 (15):	
3 (15):	
4 (8):	
5 (30):	
6 (20):	
TOTAL (100):	

Problem 1. (12 points)

Explain casts:

1. NUM_A is implicitly casted into a float, multiplied by NUM_B and assigned to ANS.
2. NUM_B is explicitly casted into an int and then multiplied by NUM_A, and this product is implicitly casted into a float to be assigned to ANS
3. NUM_A is explicitly casted into a float and then multiplied by B, and this product is assigned to ANS
4. NUM_A is implicitly casted into a float and assigned to ANS.

Problem 2. (15 points)

C characters

```
// This function returns:
//       1 if the char input is upper case
//      -1 if the char is lowercase
//       0 if it is neither.
int
character_case(char input) {
  if (input >= 'A' && input <= 'Z')
        return 1;
  else if (input >= 'a' && input <= 'z')
        return -1;
  else
        return 0;
}
```

Problem 3. (15 points)

Convert for loop to while loop:

```
// The following code asks the user for 10 integers
// and prints out their sum.
int i, num, sum = 0;

i = 0;
while (i < 10) {
      printf("Please enter a number:");
      scanf("%d",&num);
      sum = sum + num;
      i++;
}

printf("%d",sum);
```

Problem 4. (8 points)

At least 23.

Problem 5. (30 points)

Output:

```
x=    0
f1= 120
x=    1
*
x=    2
f3=  10
```

Problem 6. (20 points)

Bugs:

- Function declared to return `int`, but return statement missing
- Size of `s[]` argument not provided -- may overrun
- `length` should be `i-1` after the loop (`i` is one position after the index of `c` after each loop iteration)
- For even–length strings, the for loop swaps the middle characters twice, restoring the initial order; the termination condition for the loop should be `i < length/2`
- A temporary variable is needed for swapping two string positions; as provided, the code discards the characters in the second half of the string.

```
int
reverse_string(char s[])
{
      int i, length;

      // Determine the string length
      i = 0;
      c = s[0];
      while (c != '\0') {
            c = s[i++];
      }

      length = i;

      for (i=0; i <= length/2; i++) {
            // Swap characters at opposing ends of the string
            s[length-i-1] = s[i];
            s[i] = s[length-i-1];
      }
}
```