Question 10-1: Example 10 -4 generates the answer 47 instead of the expected answer 144. Why? (See the hint below.)

Example 10-4. first/first.c

```
#include <stdio.h>

#define FIRST_PART          7
#define LAST_PART 5
#define ALL_PARTS  FIRST_PART + LAST_PART

int main() {
        printf("The square of all the parts is %d\n", ALL_PARTS * ALL_PARTS);
        return (0);
}
```

Question 10-4: Example 10 -7 is supposed to print the message "Fatal Error: Abort" and exit when it receives bad data. But when it gets good data, it exits. Why?

Example 10-7. die/die.c

```
#include <stdio.h>
#include <stdlib.h>

#define DIE \
        fprintf(stderr, "Fatal Error:Abort\n");exit(8);

int main() {
        /* a random value for testing */
        int value;

        value = 1;
        if (value < 0)
                DIE;

        printf("We did not die\n");
        return (0);
}
```

Question 10-5: What does Example 10-8 output? Try running it on your machine. Why did it output what it did? Try checking the output of the preprocessor.

Example 10-8. sqr/sqr.c

```
#include <stdio.h>

#define SQR(x) (x * x)

int main(){
        int counter; /* counter for loop */

        for (counter = 0; counter < 5; ++counter) {
                printf("x %d, x squared %d\n",counter+1, SQR(counter+1));
        }

        return (0);
}
```

Question 10-6: Why will Example 10-9 not produce the expected output? By how much will the counter go up each time?

Example 10-9. sqr-i/sqr-i.c

```
#include <stdio.h>

#define SQR(x) ((x) * (x))

int main(){
        int counter; /* counter for loop */
        Counter = 0;

        while (counter < 5)
                printf("x %d square %d\n", counter, SQR(++counter));
        return (0);
}
```

Question 10-7: Example 10-10 tells us that we have an undefined variable number, but our only variable name is counter .

Example 10-10. rec/rec.c

```
#include <stdio.h>
#define RECIPROCAL (number) (1.0 / (number))

int main() {
        float counter; /* Counter for our table */

        for (counter = 1.0; counter < 10.0;counter += 1.0) {
                printf("1/%f = %f\n", counter, RECIPROCAL(counter));
        }
        return (0);
}
```

Exercise 10-1: Write a macro that returns TRUE if its parameter is divisible by 10 and FALSE otherwise.

Exercise 10-2: Write a macro is_digit that returns TRUE if its argument is a decimal digit.

Exercise 10-3: Write a second macro is_hex that returns true if its argument is a hex digit (0-9, A-F, a-f). The second macro should reference the first.

Exercise 10-4: Write a preprocessor macro that swaps two integers. (For the real hacker, write one that does not use a temporary variable declared outside the macro.)