1. What are the **FIVE STEPS** that describe the execution of a `for` loop?

   (a) 

   (b) 

   (c) 

   (d) 

   (e)
2. For each of these kinds of statements, mark CAN if it can appear in the body of a `for` loop, and mark CANNOT if it cannot appear in the body of a `for` loop. EXPLAIN.

   (a) A named constant declaration

   (b) A variable declaration

   (c) A `printf` statement

   (d) A `scanf` statement

   (e) An assignment statement

   (f) A `exit` statement

   (g) An `if` block

   (h) A `while` loop

   (i) A `for` loop
3. Convert the following while loop into a for loop.

```
count = initial_value;
while (count <= final_value) {
    printf("count = %d\n", count);
    count += stride;
} /* while (count <= final_value) */
```

4. Convert the following for loop into a while loop.

```
for(count = 1; count <= n; count++) {
    n_factorial *= count;
} /* for count */
```
5. What is the **OUTPUT** of each of these programs? If you aren’t confident of an answer, type in, compile and run the program to test it.

(a) 
```c
#include <stdio.h>
int main ()
{ /* main */
    int count;
    int sum = 0;

    for (count = 1; count <= 10; count++) {
        sum = sum + count;
    } /* for count */
    printf("sum = %d\n", sum);
    return 0;
} /* main */
```

(b) 
```c
#include <stdio.h>
int main ()
{ /* main */
    int count;
    int sum = 0;

    for (count = 1; count <= 10; count += 2) {
        sum = sum + count;
    } /* for count */
    printf("sum = %d\n", sum);
    return 0;
} /* main */
```

(c) 
```c
#include <stdio.h>
int main ()
{ /* main */
    int count;
    int product = 1;
    for (count = 1; count <= 15; count += 5) {
        product = product * count;
    } /* for count */
    printf("product = %d\n", product);
    return 0;
} /* main */
```
(d) #include <stdio.h>
int main ()
{ /* main */
    int count;
    int product = 1;

    for (count = 1; count <= 16; count += 5) {
        product = product * count;
    } /* for count */
    printf("product = %d\n", product);
    return 0;
} /* main */

(c) #include <stdio.h>
int main ()
{ /* main */
    const int lower_bound = 1;
    const int upper_bound = 17;
    const int stride = 5;
    int count;
    int product = 1;

    for (count = lower_bound;
        count <= upper_bound; count += stride) {
        product = product * count;
    } /* for count */
    printf("product = %d\n", product);
    return 0;
} /* main */
6. The combination of \( n \) items taken \( r \) at a time, denoted \( nC_r \) and pronounced “\( n \) choose \( r \),” is the number of ways in which you can choose a subset of \( r \) items from a set of \( n \) items:

\[
{n \choose r} = \frac{n \cdot (n-1) \cdot (n-2) \cdots (n-r+1)}{1 \cdot 2 \cdot 3 \cdots r}
\]

For example, if you have a standard 52 card poker deck (assuming no jokers and nothing wild), the number of possible 5 card poker hands is “52 choose 5:"

\[
{52 \choose 5} = \frac{52 \cdot 51 \cdot 50 \cdot 49 \cdot 48}{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5} = \frac{311,875,200}{120} = 2,598,960
\]

Write a C program that takes as its input \( n \), the total number of items available (for example, 52 cards in a deck), and \( r \), the size of the subgroup to be created (for example, 5 cards in a poker hand), and outputs \( nC_r \). (For this homework question, you do not need to include comments if you don’t want, and you may use numeric literal constants in the body of the program.)

If you use ANY resources other than Dr. Neeman, the TAs (Glose, Ivanov, Mirza, Narasimhan, Smeltzer), the course textbook or the materials posted on the course webpage, you MUST reference them on the quiz. THIS INCLUDES CLASSMATES, FRIENDS, PROFESSORS, ONLINE RESOURCES, ETC.