|  |
| --- |
| 1. What output does the following program fragment produce?   1 2 4 8 16 32 64 128 |
| 1. What output does the following program fragment produce?   9384 938 93 9 |
| 1. What output does the following for statement produce?   5 4 3 2 |
| 1. Which one of the following statements is not equivalent to the other two (assuming that the loop bodies are the same)?   (c) |
| 1. Which one of the following statements is not equivalent to the other two (assuming that the loop bodies are the same)?   (c) |
| 1. Translate the program fragment of Exercise 1 into a single for statement.   for (i = 0; i < 128; i \*= 2) |
| 1. Translate the program fragment of Exercise 2 into a single for statement.   for (i = 9384; i > 0; i /= 10) |
| 1. What output does the following for statement produce?   10 5 3 2 1 1 1 1 1 1 ... |
| 1. Translate the for statement of Exercise 8 into an equivalent while statement. You will need one statement in addition to the while loop itself.   i = 10;  while (i >= 1) {  printf(“%d ”, i++);  i /= 2;  } |
| 1. Show how to replace a continue statement by an equivalent goto statement.   for ( ... ) {  ...  continue --> goto END\_OF\_THE\_LOOP\_BODY;  ...  END\_OF\_THE\_LOOP\_BODY:  } |
| 1. What output does the following program fragment produce?   20 |
| 1. This loop isn’t very efficient. It’s not necessary to divide n by all number between 2 and n – 1 to determine whether it’s prime. In fact, we need only check divisors up to the square root of n. Modify the loop to take advantage of this fact.   for (d = 2; (d \* d) < n; d++)  if (n % d == 0)  break; |
| 1. Rewrite the following loop so that its body is empty:   for (n = 0; m > 0; n++, m /= 2) ; |
| 1. Find the error in the following program fragment and fix it.   if (n % 2 == 0) ~~;~~  printf(“n is even \n”); |