Loop structures and Booleans

Read: Chapter 8 from textbook

Practice problems

1. Do all problems in the textbook.
2. Do Discussion questions 1, 2, 3 on p. 261-262.
3. Check the truth value of the following expression, when you put parentheses around parts of it (try all possible ways of parenthesization):
   not 3=>4 and 5.5==5.5 or 6^2<4*8 and not False
4. Write a program that checks whether an integer n provided by the user is prime (divisible exactly only by 1 and itself).
5. Write a program that checks whether a positive integer n provided by the user is even or odd. In case it is odd, check whether n-1=n1+n2 for two prime integers n1,n2.
6. Write a loop that prints all possible combinations of True/False for two boolean variables A,B
   (Hint: For a single variable A, you get all possibilities by going through the list [True, False]. For two variables A,B, you do the same for A, but for each value of A you also go through all possible values of B; this can be done with nested for-loops.)
7. Write a program that outputs the truth table of the following expression:
   ((not A and B) or C) and not False
   Output the truth table like in the book in p.247 (your columns should look nice). (Hint: Nested for-loops as in (5) will get you through all combinations of A,B,C; all you have to do in the <body> is print out the current values of A,B,C and the expression.)
8. Write a search function that is defined as follows:

   def search(x, num, low, high)

   This function takes four values for its parameters: x is a number you want to find in a list of numbers num between positions low and high (both inclusive); if the number is found, the function returns the position in num where the number is, otherwise it should return -1. For example:

   >>> search(3, [4,7,2,3,6,9], 2, 5)
   3
   >>> search(3, [4,7,2,3,6,9], 0, 2)
   -1

   Initially, your implementation doesn't make any provision for nonsensical data (e.g., low is negative) and other contingencies; change your code to return -1 even if such exceptions happen.