Assignment on Number

Q1. Write a program to create an output file containing a customized loan amortization table. Your program will prompt the user to enter the amount borrowed(the principal), the annual interest rate, and the number of payments (n). To calculate the monthly payment, it will use the formula

$$payment = \frac{ip}{1-(1+i)^{-n}}$$

his payment must be rounded to the nearest cent. After the payment has been rounded to the nearest cent, the program will write to the output file n lines showing how the debt is paid off. Each month part of the payment is the monthly interest on the principal balance, and the rest is applied to the principal. Because the payment and each month's interest are rounded, the final payment will be a bit different and must be calculated as the sum of the final interest payment and the final principal balance. Here is a sample table for a $Rs.1000$ loan borrowed at a 9% annual interest rate and paid back over 6 months.

Principal	$1000.00	Payment	$171.07
Annual interest	9.0%	Term	6 months

Payment	Interest	Principal	Balance
1	7.50	163.57	836.43
2	6.27	164.80	671.63
3	5.04	166.03	505.60
4	3.79	167.28	338.32
5	2.54	168.53	169.79
6	1.27	169.79	0.00
Final payment	$171.06		

Q2. Newton's method is one of the better choice for find the root, usually converges to a solution even faster than the bisection method, if it converges at all. Newton's method starts with an initial guess for a root, x_0 , and then generates successive approximate roots $x_1, x_2, ..., x_j, x_{j+1}, ...$,using the iterative formula

$$x_{j+1} = x_j - \frac{f(x_j)}{f'(x_j)}$$

where $f'(x)$ is the derivative of function f evaluated at $x = x_j$. The formula generates a new guess, x_{j+1}, from a previous one, x_j . Sometimes Newton's method will fail to converge to a root. In this case, the program should terminate after many trials, perhaps 100.

Write a program that uses Newton's method to approximate the nth root of a number to six decimal places. If $x^n = c$, then $x^n - c = 0$. Finding a root of the second equation will give you $\sqrt[n]{c}$. Test your program on $\sqrt{2}$, $\sqrt[3]{7}$, $\sqrt[3]{-1}$. Your program could use $c/2$ as its initial guess.

Q3. Create a program that reads an unspecified number of integer arguments from the command line and adds them together. For example, suppose that you enter the following:

java Adder 1 3 2 10
The program should display 16 and then exit. The program should display an error message if the user enters only one argument.

Q4. Create a program that is similar to the previous one but has the following differences:

Instead of reading integer arguments, it reads floating-point arguments. It displays the sum of the arguments, using exactly two digits to the right of the decimal point.
For example, suppose that you enter the following:

java FPAdder 1 1e2 3.0 4.754
The program would display 108.75. Depending on your locale, the decimal point might be a comma (,) instead of a period (.).

Q5. Write a java program which read number from user and check is this number is palindrome number or not.

Q6. prints a table of Fahrenheit temperatures and the corresponding Celsius temperatures, rounded up-to two digits after decimal point.

Q7. Write a program to find a palindromic number. A palindromic number is a number if the sum of the number and reverse of that number is palindrome number.
Example: input 72
rev 27
sum 99 is a palindrome.
input 142
rev 241
sum 383 palindrome

Note: your program take any number and find its palindromic number.