Q.1 Number Conversions

Please show the steps involved.

a) Convert binary 100101111010 to octal and to hexadecimal.
   Ans: Octal: 100 = 4, 101 = 5, 111 = 7, 010 = 2, 100101111010_2 = **4572_8**
   Hexadecimal: 1001 = 9, 0111 = 7, 1010 = A and thus, 100101111010_2 = **$97A_{16}$**

b) Convert hexadecimal 3A7D to binary.
   Ans: 3 = 0011, A = 1010, 7 = 0111, D = 1101, thus, $3A7D_{16}$ = **11101001111101_2**

c) Convert hexadecimal 765F to octal. [***Hint***: First convert 765F to binary, then convert that binary number to octal.]
   Ans: Hexadecimal to Binary: 7 = 0111, 6 = 0110, 5 = 0101, F = 1111 => $765F_{16}$ = 111011001011111_2
   Binary to Octal: 111 = 7, 011 = 3, 001 = 1, 011 = 3, 111 = 7, thus, $765F_{16}$ = **73137_8** in Octal

d) Convert binary 1011110 to decimal.
   Ans: 1011110_2 = 2^6 + 2^4 + 2^3 + 2^2 + 2^1 = **94_{10}**

e) Convert octal 426 to decimal.
   Ans: 426_8 = 4*8^2 + 2*8^1 + 6*8^0 = **278_{10}**

f) Convert hexadecimal FFFF to decimal.
   Ans: $FFFF_{16}$ = 15*16^3 + 15*16^2 + 15*16^1 + 15*16^0 = **65535_{10}**

g) Convert decimal 299 to binary, to octal and to hexadecimal.
Ans: Binary: 299_{10} = 256 + 32 + 8 + 2 + 1 = **100101011_2**
Octal: 100 = 4, 101 = 5, 011 = 3, => **453_8**
Hexadecimal: 0001 = 1, 0010 = 2, 1011 = B, => **$12B_{16}$**
299_{10} = **100101011_2** = **453_8** = **$12B_{16}$**

h) Convert hexadecimal FACE to binary.
Ans: F = 1111, A = 1010, C = 1100, E = 1110, $FACE_{16}$ = **1111101011001110_2**

Q.2 IEEE 754

Represent 14.75 in its IEEE 754 single precision floating point representation. Put your answer in hexadecimal.

Ans: Converting 14.75 into binary: 14 = 1110 and 0.75 = .11 => 14.75 = 1.11011 * 2^3

sign = positive = 0, biased exponent = 127+3 = 10000010, mantissa = 11011 and fill 0 til 32bits

Therefore, the IEEE-754 single point representation of 14.75: **0 10000010 11011000000000000000000**

In hexadecimal: (0100=4 0001=1 0110=6 1100=C) => **416C0000**

Q.3 Dangling-else Problem

4.23 ***(Dangling-`else` Problem)*** C++ compilers always associate an `else` with the immediately preceding `if` unless told to do otherwise by the placement of braces (`{` and `}`). This behavior can lead to what is referred to as the **dangling-`else` problem**. The indentation of the nested statement

```
if (x > 5)
   if (y > 5)
      cout << "x and y are > 5";
   else
      cout << "x is <= 5";
```

appears to indicate that if `x` is greater than `5`, the nested `if` statement determines whether `y` is also greater than `5`. If so, the statement outputs the string `"x and y are > 5"`. Otherwise, it appears that if `x` is not greater than `5`, the `else` part of the `if`…`else` outputs the string `"x is <= 5"`. Beware! This nested `if`…`else` statement does *not* execute as it appears. The compiler actually interprets the statement as

```
if (x > 5)
   if (y > 5)
      cout << "x and y are > 5";
   else
      cout << "x is <= 5";
```

in which the body of the first `if` is a *nested* `if`…`else`. The outer `if` statement tests whether `x` is greater than `5`. If so, execution continues by testing whether `y` is also greater than `5`. If the second condition is *true*, the proper string—`"x and y are > 5"`—is displayed. However, if the second condition is *false*, the string `"x is <= 5"` is displayed, even though we know that `x` is greater than `5`. Equally bad, if the outer `if` statement's condition is *false*, the inner `if`…`else` is skipped and nothing is displayed. For this exercise, add braces to the preceding code snippet to force the nested `if`…`else` statement to execute as it was originally intended.

Q3 Ans:

```
if (x > 5){
    if (y > 5) cout << "x and y are > 5" << endl;
}else{
    cout << "x is <= 5" << endl;
}
```

Q.4 Dangling-else Problem (cont.)

4.24 ***(Another Dangling-`else` Problem)*** Based on the dangling-`else` discussion in **Exercise 4.23**, state the output for each of the following code snippets when `x` is `9` and `y` is `11` and when `x` is `11` and `y` is `9`. We eliminated the indentation from the following code to make the problem more challenging. [*Hint:* Apply indentation conventions you've learned.]

a.

```
if (x < 10)
if (y > 10)
cout << "*****" << endl;
else
cout << "#####" << endl;
cout << "$$$$$" << endl;
```

b.

```
if (x < 10)
{
if (y > 10)
cout << "*****" << endl;
}
else
{
cout << "#####" << endl;
cout << "$$$$$" << endl;
}
```

Q4a. When x == 9 and y == 11

Output: *****

$$$$$

When x == 11 and y == 9

Output: $$$$$

b. When x == 9 and y == 11

Output: *****

When x == 11 and y == 9

Output: #####

$$$$$

Q.5 Dangling-else Problem (cont.)

(Another Dangling-else Problem) Based on the dangling-else discussion in Q.3, modify the following code to produce the output shown. Use proper indentation techniques. You must not make any additional changes other than inserting braces. We eliminated the indentation from the following code to make the problem more challenging. [Note: It's possible that no modification is necessary.]

```
if ( y == 8 )
if ( x == 5 )
cout << "@@@@@" << endl;
else
cout << "#####" << endl;
cout << "$$$$$" << endl;
cout << "&&&&&" << endl;
```

a. Assuming `x = 5` and `y = 8`, the following output is produced.

```
@@@@@
$$$$$
&&&&&
```

b. Assuming `x = 5` and `y = 8`, the following output is produced.

```
@@@@@
```

c. Assuming `x = 5` and `y = 8`, the following output is produced.

```
@@@@@
&&&&&
```

d. Assuming `x = 5` and `y = 7`, the following output is produced. [*Note:* The last three output statements after the `else` are all part of a block.]

```
#####
$$$$$
&&&&&
```

Q5:

a.

```
if (y == 8){
    if (x == 5){
        cout << "@@@@@" << endl;
    }else{
        cout << "#####" << endl;
    }
    cout << "$$$$$" << endl;
    cout << "&&&&&" << endl;
}
```

b.

```
if (y == 8){
    if (x == 5){
        cout << "@@@@@" << endl;
    }else{
        cout << "#####" << endl;
        cout << "$$$$$" << endl;
        cout << "&&&&&" << endl;
    }
}
```

or

```
if (y == 8){
    if (x == 5){
        cout << "@@@@@" <<endl;
    }
}else{
    cout << "#####" <<endl;
    cout << "$$$$$" <<endl;
    cout << "&&&&&" <<endl;
}
```

c.

```
if (y == 8){
    if (x == 5){
        cout << "@@@@@" << endl;
    }else{
        cout << "#####" << endl;
        cout << "$$$$$" << endl;
    }
    cout << "&&&&&" << endl;
}
```

d.

```
if (y == 8){
    if (x == 5){
        cout << "@@@@@" <<endl;
    }
}else{
    cout << "#####" <<endl;
    cout << "$$$$$" <<endl;
    cout << "&&&&&" <<endl;
}
```

Q.6

2.29 ***(Table)*** Using the techniques of this chapter, write a program that calculates the squares and cubes of the integers from 0 to 10. Use tabs to print the following neatly formatted table of values:

```
integer square cube
0       0      0
1       1      1
2       4      8
3       9      27
4       16     64
5       25     125
6       36     216
7       49     343
8       64     512
9       81     729
10      100    1000
```

```cpp
#include<iostream>
#include<cmath>
using namespace std;

int main(){
    cout << "integer\tsquare\tcube" << endl;
    for (int i{0}; i < 11; ++i){
        cout << i << "\t" << pow(i, 2) << "\t" << pow(i, 3) << endl;
    }
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
integer square  cube
0       0       0
1       1       1
2       4       8
3       9       27
4       16      64
5       25      125
6       36      216
7       49      343
8       64      512
9       81      729
10      100     1000

Terminal will be reused by tasks, press any key to close it.
```

Q.7

2.30 ***(Body Mass Index Calculator)*** We introduced the body mass index (BMI) calculator in **Exercise 1.9**. The formulas for calculating BMI are

$$BMI = \frac{weight\ In\ Pounds \times 703}{height\ In\ Inches \times height\ In\ Inches}$$

or

$$BMI = \frac{weight\ In\ Kilograms}{height\ In\ Meters \times height\ In\ Meters}$$

Create a BMI calculator application that reads the user's weight in pounds and height in inches (or, if you prefer, the user's weight in kilograms and height in meters), then calculates and displays the user's body mass index. Also, the application should display the following information from the Department of Health and Human Services/National Institutes of Health so the user can evaluate his/her BMI:

```
BMI VALUES
Underweight: less than 18.5
Normal:      between 18.5 and 24.9
Overweight:  between 25 and 29.9
Obese:       30 or greater
```

[*Note:* In this chapter, you learned to use the `int` type to represent whole numbers. The BMI calculations when done with `int` values will both produce whole-number results. In **Chapter 4** you'll learn to use the `double` type to represent numbers with decimal points. When the BMI calculations are performed with `double`s, they'll each produce numbers with decimal points—these are called "floating-point" numbers.]

```cpp
#include<iostream>
#include<iomanip>
using namespace std;

int main(){
    cout << "BMI VALUES" << endl
         << "Underweight: less than 18.5" << endl
         << "Normal:      between 18.5 and 24.9" << endl
         << "Overwright:  between 25 and 29.9" << endl
         << "Obese:       30 or greater" << endl;

    //use -1 to check validity of input and avoid division by 0
    double weight{-1.0}, height{-1.0};
    while (weight <= 0 || height <= 0){
        cout << "\nPlease enter your weight (in kilogram) and height (in meter):" << endl;
        cin >> weight >> height;
        if (weight <= 0 || height <= 0){
            cout << "invalid input, please try again" << endl;
        }
    }
    double BMI = weight / (height*height);
    cout << "\nYour BMI is: " << fixed << setprecision(2) << BMI << endl;

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
> Executing task: /bin/bash -c ./build/Debug/outDebug <

BMI VALUES
Underweight: less than 18.5
Normal:      between 18.5 and 24.9
Overwright:  between 25 and 29.9
Obese:       30 or greater

Please enter your weight (in kilogram) and height (in meter):
69
1.78

Your BMI is: 21.78

Terminal will be reused by tasks, press any key to close it.
```