Practice Exercises - Chapter: 06

*** Exercise 6.1: Markup**

Write a program that asks the user to enter an item's wholesale cost and its markup percentage. It should then display the item's retail price. For example:

• If an item's wholesale cost is 5.00 and its markup percentage is 100%, then the item's retail price is 10.00.

• If an item's wholesale cost is 5.00 and its markup percentage is 50%, then the item's retail price is 7.50.

The program should have a function named calculateRetail that receives the wholesale cost and the markup percentage as arguments and returns the retail price of the item.

Input Validation: Do not accept negative values for either the wholesale cost of the item or the markup percentage.

Solution 6.1:

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

double calculateRetail();

int main()
{
   double RetalPrice;
   cout << "This program calculates and displays the retail
price of an item.\n";
   RetalPrice = calculateRetail();
   cout << fixed << showpoint << setprecision(2);
   cout << "The retail price of the item is $" << RetalPrice
<<endl;
   return 0;
}

double calculateRetail()
{
   double Cost,
          Markup;
   do
   {
        cout << "What is the item's wholesale cost? ";
```

```
        cin  >> Cost;
        if (Cost < 0)
        {
             cout << "Error!\n"
                  << "Wholesale cost must be a positive
number.\n";
        }
   } while (!(Cost > 0));
   do
   {
        cout << "What is the item's markup percentage? ";
        cin  >> Markup;
        if (Markup < 0)
        {
             cout << "Error!\n"
                  << "The markup percentage must be a
positive number.\n";
        }
   } while (!(Markup > 0));
   Markup /= 100;
   return Cost * (1 + Markup);
}
```

*** Exercise 6.2: Rectangle Area**

Write a program to calculate the area of a rectangle and then display the rectangle's area. The program calls the following functions, which have not been written:

• getLength – This function should ask the user to enter the rectangle's length and then return that value as a double.

• getWidth – This function should ask the user to enter the rectangle's width and then return that value as a double.

• getArea – This function should accept the rectangle's length and width as arguments and return the rectangle's area. The area is calculated by multiplying the length by the width.

• displayData – This function should accept the rectangle's length, width, and area as arguments and display them in an appropriate message on the screen.

Solution 6.2:

```
#include <iostream>
using namespace std;
double getLength();
double getWidth();
```

```
double getArea( double&, double&);
void displayData( double&, double&, double& );

int main() {
   double length,    // The rectangle's length
          width,     // The rectangle's width
          area;      // The rectangle's area

   // Get the rectangle's length.
   length = getLength();

   // Get the rectangle's width.
   width = getWidth();

   // Get the rectangle's area.
   area = getArea( length, width );

   // Display the rectangle's data.
   displayData( length, width, area );

   return 0;
}

  double length;
  cout << "Enter the rectangle's length: ";
  cin >> length;
  return length;
}

double getWidth() {
  double width;
  cout << "Enter the rectangle's width: ";
  cin >> width;
  return width;
}

double getArea( double& l, double& w ) {
  return l * w;
}
```

```
void displayData( double& l, double& w, double& a ) {
   cout << "Length: " << l << " | Width: " << w << " | Area:
" << a << '\n';
}
```

*** Exercise 6.3: Winning Division**

Write a program that determines which of a company's four divisions (Northeast, Southeast, Northwest, and Southwest) had the greatest sales for a quarter. It should include the following two functions, which are called by the main function.

• double getSales() is passed the name of a division. It asks the user for a division's quarterly sales figure, validates the input, then returns it. It should be called once for each division.

• void findHighest() is passed the four sales totals. It determines which is the largest and prints the name of the high grossing division, along with its sales figure.

Input Validation: Do not accept dollar amounts less than $0.00

Solution 6.3:

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

double getSales(string);
void findHighest(double, double, double, double);

int main()
{
   double NE, SE, NW, SW;

   cout << "\nThis program determines which of a company's
"
        << "four divisions had the greatest sales.\n";

   NE = getSales("Northeast");
   SE = getSales("Southeast");
   NW = getSales("Northwest");
   SW = getSales("Southwest");

   findHighest(NE, SE, NW, SW);
   return 0;
}
```

```
double getSales(string Division)
{
  double Sales;

  do
  {
       cout << "What is the " << Division;
       cout << " division's quarterly sales figure? ";
       cin  >> Sales;

       if (Sales < 0.00)
       cout << "Error! Invaid sales figure.\n"
            <<  "Dollar  amount  must  be  greater  than
$0.00\n";

    } while (!(Sales > 0.00));
    return Sales;
}

void findHighest(double NE, double SE, double NW, double
SW)
{
  double Hightest;

  cout << "\nThe Hightest grossing division is the ";

  if (NE > SE && NE > NW && NE > SW)
  {
       Hightest = NE;
       cout << "Northeast ";
  }
  else if (SE > NE && SE > NW && SE > SW)
  {
       Hightest = SE;
       cout << "Southeast ";
  }
  else if (NW > SE && NW > NE && NW > SW)
  {
       Hightest = NW;
```

```
        cout << "Northwest ";
    }
    else
    {
        Hightest = SW;
        cout << "Southwest ";
    }

    cout << fixed << showpoint << setprecision(2);
    cout << "division with $" << Hightest << " in sales.\n";
}
```

*** Exercise 6.4: Find the biggest number**

Given 2 numbers n and m (n<m), find a number in range from n to m such that the sum of its digits is maximum. If there are several such integers, determine the **biggest** of them. If n>m then return F.

Example:

Input: n = 50; m = 100
Output: 99

Input: n = 50; m = 98
Output: 98

Explanation:

There are two numbers with the maximum digit sum: 98 and 89.
Since 98 > 89, so 98 is the answer.

*** Exercise 6.5: Derivative of a function**

Create a function that takes numbers x and y as arguments and returns the derivative of the function f(x) = x^y.

Example:

Input: x = -3; y = 4
Output: f'(x) = -108

*** Exercise 6.6: Economical Numbers**

A number is Economical if the quantity of digits of its prime factorization (including exponents greater than 1) is equal or lower than the digits quantity of the number itself. Given an integer n, implement a function that returns from 1 to 3:

Return "**1**": "Equidigital" if the quantity of digits of the prime factorization (including exponents greater than 1) is equal to the quantity of digits of n;

Return "**2**": "Frugal" if the quantity of digits of the prime factorization (including exponents greater than 1) is lower than the quantity of digits of n;
Return "**3**": "Wasteful" if none of the two above conditions is true.
Example 1:

is_economical(14) returns 1 (Equidigital)
//The prime factorization of 14 (2 digits) is [2 x 7] (2 digits).
//Exponents equal to 1 are not counted.

Example 2:

is_economical(125) returns 2 (Frugal)
//The prime factorization of 125 (3 digits) is [5^3] (2 digits).
//Notice how exponents greater than 1 are counted.

Example 3:

is_economical(1024) returns 2 (Frugal)
//The prime factorization of 1024 (4 digits) is [2^10] (3 digits).

Example 4:

is_economical(30) returns 3 (Wasteful)
//The prime factorization of 30 (2 digits) is [2 x 3 x 5] (3 digits).

*** Exercise 6.7: Candy Parcel**

We need to assemble a parcel of ordered candy. We got two types of candies:

- Small parcel (2 candies each parcel)
- Big parcel (5 candies each parcel)

Create a function that takes three parameters: Number of small available candies nSmall, number of big candies available nBig and desired amount of the final parcel order.
The function should return the required number of small candies to achieve the goal. The function should return -1 if the goal cannot be achieved by any possible combinations of candies.
Note:

First use of big parcel that are available to achieve the desired goal. And only then should you proceed to use the small parcel.
You can't break candies into small pieces.

Example:

CandyParcel(4, 1, 13) returns 4
// 4 small parcel = 8 candies
// 1 big parcel = 5 candies
// 8 + 5 = 13 candies
// Required number of small candies = 4

CandyParcel (4, 1, 14) returns -1

// You cannot make any combination to reach 14.

CandyParcel (2, 1, 7) returns 1
// 1 big parcel = 5 candies
// 1 small parcel = 2 candies
// 5 + 2 = 7
// Required number of small candies = 1

*** Exercise 6.8: Reorder the digits**

Create a function that reorders the digits of x (x has 4 digits) based on ascending (asc) or descending (desc) order.

Example:

Input: X = 1543
Ouput: ASC(x) = 1345
DESC(x) = 5421

*** Exercise 6.9: Primorial of a Number**

In mathematics, primorial, denoted by "#", is a function from natural numbers to natural numbers similar to the factorial function, but rather than successively multiplying positive integers, the function only multiplies prime numbers.

Create a function that takes an integer n (n > 1) and returns its primorial.

Explanation:

n = 6# signifies the product of the first 6 primes

Examples:

primorial(1) returns 2
primorial(2) returns 6
primorial(6) returns 30030

*** Exercise 6.10: Polydivisible Numbers**

Mubashir was reading about Polydivisible Numbers on Wikipedia.

In mathematics a Polydivisible Number (or magic number) is a number in a given number base with digits abcde... that has the following properties:

- Its first digit a is not 0.
- The number formed by its first two digits ab is a multiple of 2.
- The number formed by its first three digits abc is a multiple of 3.
- The number formed by its first four digits abcd is a multiple of 4.
- etc.

Create a function which takes an integer n and returns true if the given number is a Polydivisible Number and false otherwise.

Example:

Input: 345654
Output: 345654 is Polydivisible number.

Explanation:

The first digit of the number is non-zero.
The number formed by the first 2 digits (34) is divisible by 2.
The number formed by the first 3 digits (345) is divisible by 3.
The number formed by the first 4 digits (3456) is divisible by 4.
The number formed by the first 5 digits (34565) is divisible by 5.
The number formed by the first 6 digits (345654) is divisible by 6.

*** Exercise 6.11: Find Area of Circle**

Program to find Area of Circle using Pointer

Example:

Input: 5
Output: 78.5

*** Exercise 6.12: Find the factorial**

Program to find the factorial using pointers

Example:

Input: 6
Output: 720

*** Exercise 6.13: Swap Number**

Program for Swap Numbers Using Pointers

Example:

Input: A = 20
B = 40
Output: A = 40
B = 20

*** Exercise 6.14: Overloading function: Add two parameters of different data types**

Create functions which find the sum of two numbers of type int, float and double. The overloaded function add() takes two parameters of different data types.

Example:

Input:
Int: x= 25; y=50;
Float: a= 12.65; b=116.50;
Double: m= 1234.567; n=2468.357;
Output:
Sum int: 75

Sum float: 129.15
Sum double: 3702.92