1. Complete the function IsPrime, which returns true if the integer number given is a prime number and false otherwise. A prime Number can be divided evenly only by 1, or itself. And it must be a whole number greater than 1. Example: 5 can only be divided evenly by 1 or 5, so it is a prime number. But 6 can be divided evenly by 1, 2, 3 and 6 so it is NOT a prime number.

```
bool IsPrime (int number) {
    bool prime = true;
    int m = 2;
    while(m < number) {
        if(number%m == 0){
            prime = false;
        }
        m = m + 1;
    }
    return prime;
}
```

2. Write the function AllPrime, which prints all the prime numbers less than equal to the given integer value and does not return a value . A prime Number can be divided evenly only by 1, or itself. And it must be a whole number greater than 1. Example: 5 can only be divided evenly by 1 or 5, so it is a prime number. But 6 can be divided evenly by 1, 2, 3 and 6 so it is NOT a prime number.

```
void AllPrime(int n) {
    int i = 2;
    while(n > i) {
        int m = 2;
        bool prime = true;
        // Test to determine if number is prime
        while(m<i){
            if(i%m == 0) {
                prime = false;
            }
            m++;
        }
        // Handle test results
        if(prime){
            cout << i << "␣";
        }
        i++;
    }
}
```

3. Write a function, CountVowels, that takes a string as input and count the number of vowels (a,e,i,o,u either upper or lower case) alphabetic characters. The function returns an integer value for the number of vowels.
   Notes:

   - You do not need to write the main(), the include, or namespace commands, you only need to write the function
   - Use (int)str.length() to get the length of the string variable where str is a variable name.
   - Calling function with a string "One1, Two2", should return 3.

```
int CountVowels (string input){
    int occurrences = 0;
    string one = "a";
```

```
    while(input.find("a")!=string::npos) {
        input.replace(input.find("a"),one.length(),"-");
        occurrences++;
    }
    while(input.find("A")!=string::npos) {
        input.replace(input.find("A"),one.length(),"-");
        occurrences++;
    }
    while(input.find("e")!=string::npos) {
        input.replace(input.find("e"),one.length(),"-");
        occurrences++;
    }
    while(input.find("E")!=string::npos) {
        input.replace(input.find("E"),one.length(),"-");
        occurrences++;
    }
    while(input.find("i")!=string::npos) {
        input.replace(input.find("i"),one.length(),"-");
        occurrences++;
    }
    while(input.find("I")!=string::npos) {
        input.replace(input.find("I"),one.length(),"-");
        occurrences++;
    }
    while(input.find("o")!=string::npos) {
        input.replace(input.find("o"),one.length(),"-");
        occurrences++;
    }
    while(input.find("O")!=string::npos) {
        input.replace(input.find("O"),one.length(),"-");
        occurrences++;
    }
    while(input.find("u")!=string::npos) {
        input.replace(input.find("u"),one.length(),"-");
        occurrences++;
    }
    while(input.find("U")!=string::npos) {
        input.replace(input.find("U"),one.length(),"-");
        occurrences++;
    }

    return occurrences;
}
```

4. Write a function, **GolfName**, that is given an integer value for the number of stokes above or below par. The function will return a string value for the golf name of the that score.

   Your function will return the following values depending on the number of surfaces to the shape:

   - -2 - "EAGLE"
   - -1 - "BIRDIE"
   - 0 - "PAR"
   - 1 - "BOGIE"
   - 2 - "DOUBLE BOGIE"
   - 3 - "TRIPLE BOGIE"
   - otherwise return "NO COMMENT"

```
string GolfName(int strokes){
    string name = "";
    if(strokes == -2){
        name = "EAGLE";
    } else if (strokes == -1) {
        name = "BIRDIE";
    } else if (strokes == 0) {
        name = "PAR";
    } else if (strokes == 1) {
        name = "BOGIE";
    } else if (strokes == 2) {
        name = "DOUBLE BOGIE";
    } else if (strokes == 3) {
        name = "TRIPLE BOGIE";
    } else {
        name = "NO COMMENT";
    }
    return name;
}
```

5. Write a function named **PaintHouse** to calculate the cost to paint a house. The function will take an integer for the width, the depth, and the number of windows/doors in the house. The function returns a floating point value for the total cost of the work.

   Note: You do not need to write the main(), the include, or namespace commands, you only need to write the function.

   Calculate the cost by calculating the linear feet in the perimeter of the house times the cost per foot, plus the cost of the work for each window or door times number of windows/doors.

   Calculate the paint cost by multiplying the perimeter by $8:
   paint house cost = (width + depth) * 2 * 8.
   The cost to paint the windows and doors is $6.75:
   paint openings = number of windows_or_doors * 6.75
   The return value is paint house cost + pain openings

```
float PaintHouse(int width, int depth, int numWindows) {
    float paintCost = (width+depth)*2*8;
    float windowsCost = numWindows * 6.75;
    return paintCost + windowsCost;
}
```

6. Write a function named **TuitionCost** to calculate the cost of a student's tuition. The function will take an integer for the number of units and a floating point value for the cost per unit. The function will return a floating point value for the total cost of tuition.

   Calculate the cost as units times the per unit cost:
   cost = units * cost per unit

   If the number of units is greater than 16, then cost is calculated as:
   cost = 16 * cost per unit + ((number of units over 16) * (cost per unit * 1/2))

```
float TuitionCost(int numUnits, float unitCost) {
    float cost = 0;
    if(numUnits < 16){
        cost = numUnits * unitCost;
    } else {
        cost = 16 * unitCost + ((numUnits - 16) * (unitCost * 0.5));
    }
```

```
    return cost;
}
```

7. Write a function, **GasBill**, that is given integer value for units used. The function will return a floating point value for total cost.

   Your function will calculate a cost using the following conditions as the cost of the gas depends on how much is used. First 100 units is charged at $1.23 per unit, the next 100 units are $1.14 per unit, any units above 200 are charged at $1.08 per unit.

   For example: if 244 units were used, 100*1.23 + 100*1.14 + 44*1.08 would calculate the cost. If only 87 units were used, cost would be 87*1.23.

```
float GasBill(int units) {
float total = 0;
if(units > 100) {
    total = total + (1.23 * 100);
    units = units - 100;
} else {
    total = total + (1.23 * units);
    units = 0;
}
if(units > 100) {
    total = total + (1.14 * 100);
    units = units - 100;
} else if (units > 0){
    total = total + (1.14 * units);
    units = 0;
}
if(units > 0) {
    total = total + (1.08 * units);
}
return total;
```

8. Write a function, **SumValues** that take three integer parameters, N, divisor_one, and divisor_two. The function sums all values from 1 up to and including N that are evenly divisible by either divisor_one or divisor_two, or both divisors. The function returns an integer value.
   Notes:
   Calling function with 10, 3, and 5, should return the sum of values from 1 to 10 that can be evenly divided by 3 or 5 (3, 5, 6, 9, 10), which results in a value of 33.

```
int SumValues(int N, int divisor_one, int divisor_two) {
    int start = 1;
    int total = 0;
    while(start <= N){
        if(start%divisor_one == 0){
            total = total + start;
        } else if (start%divisor_two == 0){
            total = total + start;
        }
        start++;
    }
    return total;
}
```

9. Write a function, **SumDigits**, that takes a string as input and sums the values for each digit characters (0 through 9). The function returns an integer value for sum.

Notes:
You do not need to write the main(), the include, or namespace commands, you only need to write the function. Use (int)str.length() to get the length of the string variable. Calling function with string "123", should return 6 (1+2+3).

```
int SumDigits(string input){
    int total = 0;
    while(input.length() > 0){
        char num = input[input.length()-1];
        total = total + num-'0'; // doing -'0' to a char will turn it into an
            ↪ int
        input = input.substr(0, input.length()-1);
    }
    return total;
}
```