1. The following function prints a full numeric-valued pyramid. Debug the code to fix all the compilation and run-time errors, so that the code generates the desired output. For instance, when the num value passed to the function is 4, the output would look like the following.

```
1

 2

3 3

 4 4
```

```
void fullPyramid(int num){

r (int i=0; i <= num; i+)

  for (int j=0; j =< num; j++
  {

    if (j > num-i-1)
    {
       cout << i << "␣";
    }
    else
    {
       cout << "␣";
    }
  }
  cout << endl
}

void fullPyramid(int num){
if(num > 1){
  for (int i=1; i < num+1; i++){
    cout << "␣";
      for (int j=0; j < num; j++){
        if (j >= num-i){
           cout << i << "␣";
        }else{
           cout << "␣";
        }
      }
      cout << endl;
    }
}
}
```

2. The following function prints a left half-pyramid populated by the a repeating sequence 0-9 values (see example below). Debug the code to fix all the compilation and run-time errors, so that the code generates the desired output. For instance, when the num value passed to the function is 6, the output would look like the following.

```
0
2
5
9
4
0
```

```
void leftHalfPyramid(int num) {
t counter = 0;
r (int i=0, i <= num, i++)
{
for (int j=0, j <= num, j++)
{
  cout << (i+j) / 10 << endl;
}
cout << "␣";

unter =+ 1;

void leftHalfPyramid(int num) {
counter=0;

um > 1){
r (int i=1; i < num+1; i++){
  for (int j=0; j < num; j++){
    if (j >= num-i){
        cout << counter;
        if(counter == 9){
            counter = 0;
        } else {
            counter++;
        }
    }else{
       cout << "␣";
    }
  }
  cout << endl;
}
```

3. Write a function, void StarsAndDashes(int n) to print the following pattern.The function takes in an integer value which is the number of lines in your pattern. The number of stars and dashes in each line will be same as the integer value n. The function does not return any value.

   For instance the function will print the following when n = 5:

   ***** ___ ***** ___ *****

```
void StarsAndDashes(int n) {
 star=true;

<=0){
exit;


c;
l;

l=0; l<n; l++){
for(c=0; c<n; c++){
    if(star){
        cout << "*";
    } else {
        cout << "-";
    }
}
star = !star;
cout << endl;
```

```
}
```

4. Write a function that takes in a string array and its size and returns the number of times character 'e' appeared in the whole array.

   int countCharacter(string array[ ], int size)

   Example:

   array = "elephant", "english", "elegant", "america", "united", "apple"

   return value: 8

```
int countCharacter(string array[ ], int size){
 letter = 'e';
letter_count;

int w=0; w<size; w++){
string word = array[w];
for(int c=0; c<word.length(); c++){
    char w_letter = word[c];
    if(w_letter == letter){
        letter_count++;
    }
}


rn letter_count;
```

5. Write a function countPrimes that takes in two parameters: an int array and the size of the array. The function returns the count of the number of prime elements in the array.

   int countPrimes(int array[], int size)

   Given array: 1, 2, 3, 4, 5, 7, 10, 12

   Expected return: 4

   Explanation: The primes are 2, 3, 5, and 7; consequently, the function returns a count of 4.

```
int countPrimes(int array[], int size){
prime_count = 0;
int i=0; i<size; i++){
int num = array[i];
int m = 2;
bool prime = true;

if(m>num)
    prime=false;

while(m< num){
    if(num%m == 0){
        prime = false;
    }
    m++;
}
if(prime){
    prime_count++;
}


rn prime_count;
```

6. Write a function findPrimeSum which takes in two parameters: an array and the size of the array. The function prints the sum of prime elements in the array.

   void findPrimeSum(int array[], int size)

   Given array: 1, 2, 3, 4, 5, 7, 10, 12

   Output: 17

   Explanation: The primes are 2, 3, 5, and 7; consequently, 2+3+5+7 = 17.

```
void findPrimeSum(int array[], int size){
prime_total = 0;
int i=0; i<size; i++){
int num = array[i];
int m = 2;
bool prime = true;

if(m>num)
    prime=false;

while(m< num){
    if(num%m == 0){
        prime = false;
    }
    m++;
}
if(prime){
    prime_total += num;
}


 << prime_total;
```

7. Write a function multiplicationOfArrays which takes four parameters: two arrays (X and Y) and their sizes. The function should find the sum of multiplication of each element of X with all elements of Y.

   The function should return the final sum.

   int multiplicationOfArrays(int X[], int size1, int Y[], int size2)

   Example:

   X = 2,3,4,5,6

   Y = 3,4,5

   calculation: (2*3 + 2*4 + 2*5) + (3*3 + 3*4 + 3*5) + (4*3 + 4*4 + 4*5) + (5*3 + 5*4 + 5*5) + (6*3 + 6*4 + 6*5)

   Output : 240

```
int multiplicationOfArrays(int X[], int size1, int Y[], int size2){
total = 0;
int a=0; a<size1; a++) {
for(int b=0; b<size2; b++){
    total+= X[a]*Y[b];
}

rn total;
```

8. Given an array and a starting position write a function replaceFromN, that takes an integer array 'array', the size of the array size and a starting positions 'n' as parameters and replaces the elements starting from that index onward with the sequence 1,2,3,... The function returns nothing.

void replaceFromN(int array[], int size, int n)

For example, given

array= 15,12,4,9,2,3

n =2

the function should modify array to be 15,12,1,2,3,4

```
void replaceFromN(int array[], int size, int n){
 replacing=false;
replace_count=1;

int i=0; i<size; i++){
if(i == n){
    replacing = true;
}
if(replacing){
    array[i]=replace_count;
    replace_count++;
}
```

9. Create a function named ReadCities, which takes a string input filename and string output filename as a parameters. This function returns the number of cities read from the file. If the input file cannot be opened, return -1 and do not print anything to the output file.

   Read each line from the given filename, parse the data, process the data, and print the required information to the output file.

   Each line of the file contains CITY, STATE, POPULATION, ELEVATION. Read the data and print the city with the largest population and the city with the highest elevation.

   If given the data below:

   Seattle, Wash., 668342, 429 Denver, Colo., 663862, 5883 Washington, DC, 658893, 16 Indianapolis, Ind., 848788, 797 New York, N.Y., 8491079, 13 Los Angeles, Calif., 3928864, 126 Your output file should contain:

   Largest: New York City, 8491079 Highest: Denver, 5883 You only need to write the function code. Our code will create and test your class.

```
int ReadCities(string inputfilename, string outputfilename){
ITY, STATE, POPULATION, ELEVATION
ooking for largest pop and highest elv

largest_pop = 0;
ng largest_name;
highest_elv = 0;
ng highest_name;

cities_read = 0;

ream input;
t.open(inputfilename);

input.fail()){
// File opening failed
//cout << "Input file opening failed." << endl;
return -1;
se {
// File opening suceeded
```

```
string line;
while(getline(input,line)){
    istringstream ss(line);
    int index = 0;
    string result;
    string city;
    while(getline(ss, result, ',')) {
        if(index==0){
            // City
            city = result;
        } else if(index==2){
            // Process population
            if(stoi(result)>=largest_pop){
                largest_pop = stoi(result);
                largest_name = city;
            }
        } else if(index==3){
            // Process population
            if(stoi(result)>=highest_elv){
                highest_elv = stoi(result);
                highest_name = city;
            }
        }
        index++;
    }
    cities_read++;
}
input.close();

ofstream output;
output.open(outputfilename);
if (output.fail()){
    cout << "Error writing file" << endl;
} else {
    output << "Largest: " << largest_name << ", " << largest_pop << endl;
    output << "Highest: " << highest_name << ", " << highest_elv << endl;

    output.close();
}


rn cities_read;
```

10. Create a function named CalcCost, which takes a string input filename and string output filename as a parameters. This function returns the number of entries read from the file. If the input file cannot be opened, return -1 and do not print anything to the output file. Read each line from the given filename, parse the data, process the data, and print the required information to the output file.

    Each line of the file contains PRODUCT NAME, COST PER ITEM, QUANTITY. Read and parse the data, then output the total cost for each entry.

    If given the data below:

    Seattle Coffee, 12.54, 39 Denver Mints, 1.00, 1877 Computer, 699.95, 16 Your output file should contain:

    Seattle Coffee: 489.06 Denver Mints: 1877.00 Computer: 11199.20 You only need to write the function code. Our code will create and test your class.

    ```
    int CalcCost(string inputfilename, string outputfilename){
    RODUCT NAME, COST PER ITEM, QUANTITY
    ```

```
ooking for name and multiplied values

entries_read = 0;
ng output_string = "";

ream input;
t.open(inputfilename);

input.fail()){
// File opening failed
//cout << "Input file opening failed." << endl;
return -1;
se {
// File opening suceeded
string line;
while(getline(input,line)){
    string name = "";
    double cost = 0;
    int quantity = 0;

    istringstream ss(line);
    int index = 0;
    string result;
    while(getline(ss, result, ',')) {
        if(index==0){
            name = result;
        } else if(index==1){
            result = result.substr(1, result.length()-1);
            cost = stod(result);
        } else if(index==2){
            result = result.substr(1, result.length()-1);
            quantity = stoi(result);
        }
        index++;
    }
    double tot = (cost*quantity);
    string num_str = to_string(tot);
    while(num_str[num_str.length()-1]=='0'){
        if(num_str[num_str.length()-2]=='.'){
            num_str = num_str.substr(0, num_str.length()-2);
            break;
        }else{
            num_str = num_str.substr(0, num_str.length()-1);
        }
    }

    output_string = output_string + name + ": " + num_str + "\n";

    entries_read++;
}
input.close();

ofstream output;
output.open(outputfilename);
if (output.fail()){
    cout << "Error writing file" << endl;
} else {
    output << output_string;
    output.close();
```

```
    }

    rn entries_read;
```