1) create an abstract class named "Shape" in C++ with Pure Virtual functions to calculate the area and Perimeter. Implement two derived classes "circle" and "Rectangle" that inherit from the "Shape" class and provide concrete implementations for the area and Perimeter calculations.

```
#include <iostream>
class shape{
Public:
    Virtual double calculate Area ()const =0;
    Virtual double calculate Perimeter ()const =0;
};
class circle : Public shape{
Private:
    double radius;
Public:
    circle (double r) : radius(r){}
```

```
double calculateArea() const override {
    return 3.14159 * radius * radius;
}

double calculatePerimeter() const override {
    return 2 * 3.14159 * radius
}
};

class Rectangle : Public Shape{
Private:
    cdouble length;
    double width;
Public:
    Rectangle(double l, double w): length(l), Width(w){}

    double calculateArea() const override {
        return length * width;
    }

    double calculatePerimeter() const override {
        return 2* (length + width)
    }
```

```
};
int Main () {
    circle circle (5.0)
    Rectangle rectangle (4.0, 6.0);
    << "Rectangle - Area: " << rectangle . calculate Area () << "; Perimeter: "
rectangle . calculate Perimeter () endl;
    return 0;
}
```

2) Write a C++ Program that includes a function to divide two Numbers Handle exceptions such as divide by zero and invalid inputs using try-catch blocks.

```
# include<iostream>

double divide (double Numerator, double denomenator ){
    if (denomenator == 0){
        throw runtime_error ("Division by zero is not allowed.");
    }
    return Numerator / denomenator;
}

int Main (){
    double Numerator, denomenator;
```

```
cout << "Enter numerator: ";
cin >> Numerator;
cout << "Enter denominator: ";
cin >> denominator;
try {
    double Result = divide (Numerator, denominator);
    cout << "Result:" << result << endl;
} catch (const exception &e){
    cerr << "Error: " << e.what() endl;
}
return 0;
}
```

3) Develop a C++ program that reads data from an input file processes it, and writes the result to an output file. Ensure proper error handling for file operations.

```
#include <iostream>
#include <fstream>
#include <string>

int main(){
        string input File name = "input.txt";
```

```
string outputFileName = "output.txt";
ifstream inputFile(inputFileName);
if (!inputFile){
    cerr << "Error: unable to open input file: " << inputFileName << endl;
    return 1;
}
ofstream outputFile(outputFileName);
if (!outputFile){
    cerr << "Error: unable to open output file: " << outputFileName << endl;
    inputFile.close();
    return 1;
}
string line;
while (getline(inputFile, line)){
    outputFile << line << endl;
}
inputFile.close();
outputFile.close();
cout << "Data has been processed and written to " << outputFileName << endl;
return 0;
}
```

4) Implement a C++ Program that Performs various operations on a vector, such as adding elements, removing duplicates, and finding the sum of all elements.

```
#include <iostream>
#include <vector>
#include <algorithm>

Void add elements (vector<int> &Vec){
int num;
    cout << "Enter elements (enter -1 to stop): ";
    while (true){
        cin >> Num;
        if (Num == -1){
          break;
        }
        Vec.push_back(Num);
    }
}

Void remove Duplicates (Vector<int> &Vec){
sort (Vec.begin(), Vec.end());
Vec.erase (unique (Vec.begin(), Vec.end()), Vec.end());
}
```

```
int findSum(const Vector<int>& vec){
    int sum = 0;
    for (int num: vec){
        sum += num;
    }
    return sum;
}
int main(){
    Vector<int> vec;
    addElements(vec);
    cout << "original Vector: ";
    for (int num : vec) {
        cout << "original Vector: ";
        for (int num : vec ) {
            cout << num << " ";
        }
        cout << endl;
        removeDuplicates(vec);
        cout << "Vector after removing duplicates: ";
        for (int num : vec ) {
            cout << num << " ";
        }
        cout << endl;
        int sum = findSum(vec);
        cout << "Sum.
```

5) Design a C++ program that defines a generic function to find the Maximum value in array of any data type (int, double, string). use this function to find the maximum of different data types.

```
#include <iostream>
#include <vector>
#include <string>

template <type name T>
T find Max (const T arr [] int size){
    if (size <=0){
        throw invalid_argument ("Array size must be
                                 greater than zero.");
    }

    T max val = arr [0];
    for (int i = 1 ; i < size; ++i){
    if (arr [i] > Max Val){
     Max val = arr [i];
    }
    }
    return Max val;
}
int main (){
```

```
int intArray[] = {10, 20, 5, 30, 15};
int intMax = findMax(intArray, 5);
cout << "Maximum integer value: " << intMax << endl;

double doubleArray[] = {3.14, 2.718, 1.618, 0.577};
double doubleMax = findMax(doubleArray, 4);
cout << "Maximum double value: " << doubleMax << endl;

string stringArray[] = {"apple", "banana", "cherry", "date"};
string stringMax = findMax(stringArray, 4);
cout << "Maximum string value: " << stringMax << endl;

return 0;
}
```