1. Write a code that performs username validation for a website. When the username is too short it should throw an exception such that it prints Too short: n (where n is the length of the given username). The final program should print Valid (if the username is valid), Invalid (if the username is invalid), or Too short: n (where is the length of the too-short username). Make necessary assumptions if required.

```
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$ cat q1.cpp
#include <bits/stdc++.h>
#include <iostream>

using namespace std;

#define MIN_LENGTH 5
#define MAX_LENGTH 15

void validate(string s) {
    if (s.length() < MIN_LENGTH || s.length() > MAX_LENGTH) {
        throw invalid_argument("Invalid string length");
    }
}

int main () {
    string username;
    cout << "Enter your username: ";
    cin >> username;
    try {
        validate(username);
    } catch (invalid_argument &e) {
        cout << "Invalid username: " << e.what() << endl;
    }
    cout << "Username: " << username << endl;
    cout << endl;
    return 0;
}maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$ ./a.out
Enter your username: namankhater
Username: namankhater

maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$ ./a.out
Enter your username: nam
Invalid username: Invalid string length
Username: nam

maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$
```

2. You are required to handle error messages while working with a small computational server that performs complex calculations. It has a function that takes 2 large numbers as its input and returns a numeric result. Unfortunately, there are various exceptions that may occur during execution. Write a program so that it prints appropriate error messages. The expected behavior is defined as follows:
    - If the compute function runs fine with the given arguments, then print the result of the function call.
    - If it fails to allocate the memory that it needs, print Not enough memory.
    - If any other standard C++ exception occurs, print Exception: S where S is the exception's error message.

- If any non-standard exception occurs, print Other Exceptions.

```
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$ cat q2.cpp
#include <iostream>
#include <bits/stdc++.h>

using namespace std;

int compute(int a, int b) {
    try {
        return a / b;
    } catch (bad_alloc &e) {
        cout << "Not enough memory" << endl;
    } catch (exception &e) {
        cout << "Exception: " << e.what() << endl;
    } catch (...) {
        cout << "Other Exceptions" << endl;
    }
}

int main () {
    int a, b;
    cout << "Enter two numbers: ";
    cin >> a >> b;
    cout << "Result: " << compute(a, b) << endl;
    return 0;
}maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$ g++ q2.cpp
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$ ./a.out
Enter two numbers: 5 10
Result: 0
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$ ./a.out
Enter two numbers: 10 5
Result: 2
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$ ./a.out
Enter two numbers: 10 0
Floating point exception (core dumped)
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$
```

3. Create a class Polar that represents the points on the plane as polar coordinates (radius and angles). Create an overloaded + operator for addition of two Polar quantities. "Adding" two points on the plane can be accomplished by adding their X coordinates and then adding their Y coordinates. This gives the X and Y coordinates of the "answer." Thus you'll need to convert two sets of polar coordinates to rectangular coordinates, add them, then convert the resulting rectangular representation back to polar. You need to use the following trigonometric formulae:

x = r*cos(a);

y = r*sin(a);

a = atan(y/x); //arc tangent

r = sqrt(x*x + y*y);

```
#include <iostream>
#include <bits/stdc++.h>

using namespace std;

class Polar {
```

```
    public:
        double radius;
        double angle;
        Polar(double r, double a) {
            radius = r;
            angle = a;
        }
        Polar operator+(Polar p) {
            double x1 = radius*cos(angle);
            double y1 = radius*sin(angle);
            double x2 = p.radius*cos(p.angle);
            double y2 = p.radius*sin(p.angle);
            double x = x1 + x2;
            double y = y1 + y2;
            double a = atan(y/x);
            double r = sqrt(x*x + y*y);
            return Polar(r, a);
        }
};

int main () {
    cout << "Enter radius and angle: (angle in degrees)" << endl;
    double r, a;
    cin >> r >> a;
    Polar p1(r, a*M_PI/180);
    cout << "\nEnter radius and angle: (angle in degrees)" << endl;
    cin >> r >> a;
    Polar p2(r, a*M_PI/180);
    Polar p3 = p1 + p2;
    cout << "\nRadius: " << p3.radius << endl;
    cout << "Angle: " << p3.angle << endl;
    return 0;
}
```

```
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$ g++ q3.cpp
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$ ./a.out
Enter radius and angle: (angle in degrees)
5 80

Enter radius and angle: (angle in degrees)
8 50

Radius: 12.581
Angle: 1.07271
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$
```

4. A file contains a list of telephone numbers in the following form:

   John 23456

   Ken 9846

The names contain only one word and the names and telephone numbers are separated by white spaces. Write a program to read a file and display its contents in two columns. The names should be left justified and the number right justified.

```
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$ cat telephone.txt
Naman 9586048530
Sushil 9586078530
Amit 9589048530
Sagar 9586848530
Mukesh 9786048530
Ishan 9586043530
Chirag 9583048530
vishal 9686048530
lakshman 9526048530
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$ cat q4.cpp
#include <iostream>
#include <bits/stdc++.h>

using namespace std;

int main () {
    string name, number;
    // read file
    ifstream fin("telephone.txt");
    cout << setw(20) << left << "Name" << setw(10) << right << "Number" << endl;
    if (fin.is_open()) {
        while (fin >> name >> number) {
            cout << setw(20) << left << name << setw(10) << right << number << endl;
        }
    }
    return 0;
}maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$ g++ q4.cpp
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab4$ ./a.out
Name                    Number
Naman               9586048530
Sushil              9586078530
Amit                9589048530
Sagar               9586848530
Mukesh              9786048530
Ishan               9586043530
Chirag              9583048530
vishal              9686048530
```