Principles of Programming Language (CS302)

Assignment - 3

U19CS012

1.) Write a code that performs **Username Validation** for a website.

You are updating the username policy on your company's internal networking platform. According to the policy, a username is considered valid if all the following constraints are satisfied:

- The username consists of 8 to 30 characters inclusive. If the username consists of less than 8 or greater than 30 characters, then it is an invalid username.
- The username can only contain alphanumeric characters and underscores (_). Alphanumeric characters describe the character set consisting of lowercase characters $[a-z]$, uppercase characters $[A-Z]$, and digits $[0-9]$.
- The first character of the username must be an alphabetic character, i.e., either lowercase character $[a-z]$ or uppercase character $[A-Z]$.

For example:

Username	Validity
Julia	INVALID; Username length < 8 characters
Samantha	VALID
Samantha_21	VALID
1Samantha	INVALID; Username begins with non-alphabetic character
Samantha?10_2A	INVALID; '?' character not allowed

Code

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

#define username_lower_limit 8
#define username_upper_limit 30

string reason;

bool valid_username(string username)
{
    int special = 0;
    int n = username.length();

    // Length of Username Constraint
    if (n < username_lower_limit || n > username_upper_limit || n <= 0)
    {
        if (n < username_lower_limit)
        {
```

```
            string t("Username length < " + to_string(username_lower_limit) +
"characters\n");
            reason = t;
        }

        if (n > username_upper_limit)
        {
            string t("Username length > " + to_string(username_upper_limit) +
"characters\n");
            reason = t;
        }

        return false;
    }

    // Starting Character Constraint
    char start_ch = username[0];
    // If first character is not {[A-Z]/[a-z]}, then username is invalid
    if (!((start_ch >= 'a' && start_ch <= 'z') || (start_ch >= 'A' && start_ch <= 'Z')))
    {
        string t("First character is not {[A-Z]/[a-z]}\n");
        reason = t;
        return false;
    }

    // All remaining Characters is not {_/[A-Z]/[a-z]/[0-9]}, then username is invalid
    for (int i = 1; i < n; i++)
    {
        char ch = username[i];
        if (!((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') || (ch >= '0' && ch <=
'9') || ch == '_'))
        {
            string t("Characters is Not from {_/[A-Z]/[a-z]/[0-9]}, i.e. Special Character
Used!\n");
            reason = t;
            return false;
        }
    }

    // If all the 3 Constraints are Satisfied, then Username is Valid
    return true;
}

int main()
{
    // Number of Usernames to be Checked
    int n;
    cout << "Enter the Number of Usernames to Check for Validity : ";
    cin >> n;
```

```
    string s[n];
    for (int i = 0; i < n; i++)
    {
        cin >> s[i];
    }
    cout << endl;

    for (int i = 0; i < n; i++)
    {
        if (valid_username(s[i]))
        {
            cout << s[i] << " -> Valid Username\n\n";
        }
        else
        {
            cout << s[i] << " -> Invalid Username\n";
            cout << "Reason : " << reason << endl;
        }
    }
    return 0;
}
```

Output

```
Enter the Number of Usernames to Check for Validity : 8
Julia
Samantha
Samantha_21
1Samantha
Samantha?10_2A
JuliaZ007
Julia@007
_Julia007

Julia -> Invalid Username
Reason : Username length < 8characters

Samantha -> Valid Username

Samantha_21 -> Valid Username

1Samantha -> Invalid Username
Reason : First character is not {[A-Z]/[a-z]}

Samantha?10_2A -> Invalid Username
Reason : Characters is Not from {_/[A-Z]/[a-z]/[0-9]}, i.e. Special Character Used!

JuliaZ007 -> Valid Username

Julia@007 -> Invalid Username
Reason : Characters is Not from {_/[A-Z]/[a-z]/[0-9]}, i.e. Special Character Used!

_Julia007 -> Invalid Username
Reason : First character is not {[A-Z]/[a-z]}
```

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.

Exceptions in C++ are run-time anomalies or abnormal conditions that a program encounters during its execution. C++ provides the following specialized keywords for exception handling:

`try` : represents a block of code that can throw an exception.

`catch` : represents a block of code that is executed when a particular exception is thrown.

Code

```
#include <iostream>
#include <vector>
// exception is for the user to inherit and define their own exceptions.
#include <exception>
// stdexcept is for catching and handling the standard exceptions
#include <stdexcept>
// [U19CS012] BHAGYA VINOD RANA
using namespace std;
typedef long long int ll;

ll complex_function(ll a, ll b)
{
    if (b < 0)
        throw invalid_argument("B is Negative\n");

    // Allocated a Vector of Size 'b' -> for bad_alloc Exception
    vector<ll> vec(b, 1);

    // For other Exception Code
    if (!(a ^ b))
    {
```

```
        throw int(a ^ b);
    }
    cout << "Output of Complex computation : ";
    return (vec[b - 1]) ^ a | b;
}

int main()
{
    ll a, b;
    cout << "Enter Two Numbers for Complex Computation : ";
    cin >> a >> b;

    // Try Catch Block for Exception Handling
    try
    {
        cout << complex_function(a, b) << "\n";
    }
    catch (bad_alloc &ba)
    {
        // Exception thrown on failure allocating memory
        cerr << "bad_alloc Exception caught : " << ba.what() << endl;
        cerr << "Not Enough Memory\n";
    }
    catch (exception &e)
    {
        cerr << "Exception : " << e.what() << endl;
    }
    catch (int e)
    {
        cerr << "Other Exceptions Code : " << e << endl;
    }

    return 0;
}
```

Output

```
PS C:\Users\Admin\Desktop\PPL_L3> cd "c:\Users\Admin\Desktop\PPL_L3\" ; if ($?) { g++ Q2.cpp -o Q2 }
Enter Two Numbers for Complex Computation : 1234 7834
Output of Complex computation : 7899
PS C:\Users\Admin\Desktop\PPL_L3> cd "c:\Users\Admin\Desktop\PPL_L3\" ; if ($?) { g++ Q2.cpp -o Q2 }
Enter Two Numbers for Complex Computation : 567891 567891
Other Exceptions Code : 0
PS C:\Users\Admin\Desktop\PPL_L3> cd "c:\Users\Admin\Desktop\PPL_L3\" ; if ($?) { g++ Q2.cpp -o Q2 }
Enter Two Numbers for Complex Computation : 102 9876543210
bad_alloc Exception caught : std::bad_alloc
Not Enough Memory
PS C:\Users\Admin\Desktop\PPL_L3> cd "c:\Users\Admin\Desktop\PPL_L3\" ; if ($?) { g++ Q2.cpp -o Q2 }
Enter Two Numbers for Complex Computation : 876 -32
Exception : B is Negative
```

No Errors Case

User Defined Exception

Nor Enough Memory Exception

Negative Number Exception

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);
```

Code

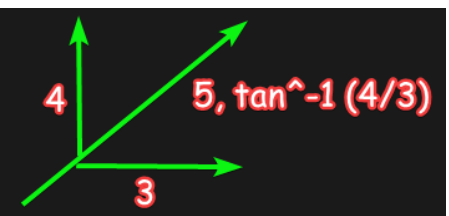
```
#include <iostream>
// for sin(), cos(), tan(), sqrt()
#include <cmath>

// [U19CS012] BHAGYA VINOD RANA

using namespace std;

const double pi = 3.14159265358979323846;

// Polar Class with '+' Operator Overloaded
class polar
{
private:
    double radius;
    // theeta in degrees
    double angle;

public:
    // Default Constructor
    polar() : radius(0.0), angle(0.0) {}

    // Parameterized Constructor
    polar(double r, double t)
    {
        radius = r;
        angle = t;
    }
```

```
    // For Displaying the Output in Polar Form (r,θ)
    void display_polar_form()
    {
        cout << "Radius (R) = " << radius << " , Theeta (θ) = " << angle << " Degree(s)\n";
    }

    // Overload the '+' Operator
    polar operator+(polar a)
    {
        // Final r' and theta θ°
        double r, t;
        double x1, x2, y1, y2, x3, y3;

        // Trigonometry F(x) -> single mandatory argument in radians.

        // x = r*cos(a);
        x1 = radius * cos(pi * angle / 180);
        x2 = a.radius * cos(pi * a.angle / 180);

        // y = r*sin(a);
        y1 = radius * sin(pi * angle / 180);
        y2 = a.radius * sin(pi * a.angle / 180);

        // Add them
        x3 = x1 + x2;
        y3 = y1 + y2;

        // Radius of Final r' = sqrt((x')*(x') + (y')*(y'));
        r = sqrt((x3 * x3) + (y3 * y3));

        // θ = tan inverse of (y'/x')
        t = atan(y3 / x3);
        // convert Back to degrees
        t = t * 180 / pi;

        return polar(r, t);
    }
};

int main()
{
    // Number of Points
    int n;

    cout << "Enter the Number of Polar Points : ";
    cin >> n;

    double rad[n], angle[n];
    polar P[n];
```

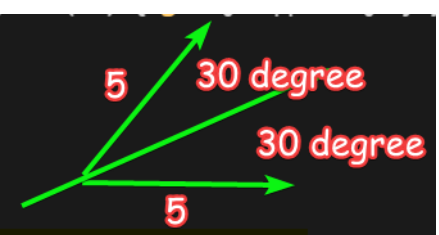
```
    for (int i = 0; i < n; i++)
    {
        cout << "Enter the Polar Co-Ordinates of Point " << i + 1 << " [ Radius, Angle(in
degree) ] : \n";
        cin >> rad[i] >> angle[i];
        P[i] = polar(rad[i], angle[i]);
    }

    polar resultant(0, 0);

    for (int i = 0; i < n; i++)
    {
        P[i].display_polar_form();
        resultant = resultant + P[i];
    }

    cout << "Resultant [Summation] Point in Polar Form : ";
    resultant.display_polar_form();

    return 0;
}
```

Output

```
Enter the Number of Polar Points : 2
Enter the Polar Co-Ordinates of Point 1 [ Radius, Angle(in degree) ] :
3 0
Enter the Polar Co-Ordinates of Point 2 [ Radius, Angle(in degree) ] :
4 90
Radius (R) = 3 , Theeta (0) = 0 Degree(s)
Radius (R) = 4 , Theeta (0) = 90 Degree(s)
Resultant [Summation] Point in Polar Form : Radius (R) = 5 , Theeta (0) = 53.1301 Degree(s)
```




```
Enter the Number of Polar Points : 2
Enter the Polar Co-Ordinates of Point 1 [ Radius, Angle(in degree) ] :
5 0
Enter the Polar Co-Ordinates of Point 2 [ Radius, Angle(in degree) ] :
5 60
Radius (R) = 5 , Theeta (0) = 0 Degree(s)
Radius (R) = 5 , Theeta (0) = 60 Degree(s)
Resultant [Summation] Point in Polar Form : Radius (R) = 8.66025 , Theeta (0) = 30 Degree(s)
```



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

John 2347038256

Ken 9841920261

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***.

Code

```
#include <iostream>
// For setw()
#include <iomanip>
// For String
#include <string>
// For File Reading
#include <fstream>
// U19CS012 [BHAGYA VINOD RANA]
using namespace std;

int main()
{
    string file_name;
    cout << "Enter the Name of the File to be Formatted [Name No] : ";
    cin >> file_name;

    // ifstream: Stream class to read from files
    ifstream ifstream_obj;

    ifstream_obj.open(file_name);

    string name;
    long long int telephone_no;

    if (ifstream_obj)
    {
        while (ifstream_obj >> name >> telephone_no)
        {
            // Adjust output to the left
            cout << left << setw(15) << name;
            // Adjust output to the right
            cout << right << setw(15) << telephone_no;
            cout << endl;
        }
        ifstream_obj.close();
    }
    else
    {
```

```
        cout << file_name << " named File Does Not Exist!\n";
    }
    return 0;
}
```

Output

sample.txt Q4.cpp

sample.txt

```
Sergio    5782374520
Raquel        8492024213
Denver            7823974501
Monica                  2748472840
Alison        8923047304
Tokyo           8349723738
Nairobi    2893072312
Rio               3894073283
Helsinki     2378133947
```

```
Enter the Name of the File to be Formatted [Name No] : sample.txt
Sergio              5782374520
Raquel              8492024213
Denver              7823974501
Monica              2748472840
Alison              8923047304
Tokyo               8349723738
Nairobi             2893072312
Rio                 3894073283
Helsinki            2378133947
```

formatted Output

If the File Does Not exist, it Reports an Error.

```
 .\Q4 }
Enter the Name of the File to be Formatted [Name No] : random.txt
random.txt named File Does Not Exist!
```

SUBMITTED BY: U19CS012

BHAGYA VINOD RANA