Recursion Problem List

Write a program that

1. Prints the numbers from 1 to n.

```cpp
#include<iostream>
using namespace std;
void func(int i, int n)
{
    if(i>n)
        return;
    else
    {
        printf("%d ", i);
        func(i+1, n);
    }
}
int main()
{
    int n; cin >> n;
    func(1, n);
    return 0;
}
```

2. Calculate the sum of numbers from 1 to n.

```cpp
#include<iostream>
using namespace std;
int func(int i, int n)
{
    if(i>n)
        return 0;
    else
    {
        int sumFromNext = func(i+1, n);
        return i + sumFromNext;
    }
}
int main()
{
    int n; cin >> n;
    int result = func(1, n);
    cout << result << endl;
    return 0;
}
```

```cpp
#include<iostream>
using namespace std;
int sum(int n)
```

```
{
    if(n==0) return 0;
    else return n + sum(n-1);
}
int main()
{
    int n; cin >> n;
    int result = sum(n);
    cout << result << endl;
    return 0;
}
```

3. Calculate the factorial of n.

```
#include<iostream>
using namespace std;
int fact(int n)
{
    if(n==0) return 1;
    else return n * fact(n-1);
}
int main()
{
    int n; cin >> n;
    int result = fact(n);
    cout << result << endl;
    return 0;
}
```

4. Calculate the sum of digits of a given number n.

```
#include<iostream>
using namespace std;
int sumOfDigits(int n)
{
    if(n==0) return 0;
    else
    {
        int lastDigit = n%10;
        n = n/10;
        return lastDigit + sumOfDigits(n);
    }
}
int main()
{
    int n; cin >> n;
    int result = sumOfDigits(n);
    cout << result << endl;
```

```cpp
    return 0;
}
```

5. Count the number of digits of a given number n.

```cpp
#include<iostream>
using namespace std;
int countDigits(int n)
{
    if(n==0) return 0;
    else
    {
        n = n/10;
        return 1 + countDigits(n);
    }
}
int main()
{
    int n; cin >> n;
    int result = countDigits(n);
    cout << result << endl;
    return 0;
}
```

6. Calculate the nth term of a Fibonacci series.

```cpp
#include<iostream>
using namespace std;
int fib(int n)
{
    if(n==1) return 0;
    else if(n==2) return 1;
    else
    {
        int friend1 = fib(n-1);
        int friend2 = fib(n-2);
        return friend1 + friend2;
    }
}
int main()
{
    int n; cin >> n;
    int result = fib(n);
    cout << result << endl;
    return 0;
}
```

7. Calculate a to the power b

```
#include<iostream>
using namespace std;
int power(int a, int b)
{
    if(b==0) return 1;
    else
    {
        int _friend = power(a, b-1);
        int result = a * _friend;
        return result;
    }
}

int main()
{
    int a, b; cin >> a >> b;
    int result = power(a, abs(b));

    if(b<0)
    {
        cout << 1.0/result << endl;
    }
    else
    {
        cout << result << endl;
    }
    return 0;
}
```

8. Print the array elements.

```
#include<iostream>
using namespace std;
void printArray(int arr[], int n, int it)
{
    if(it==n) return;
    else
    {
        cout << arr[it] << " ";
        printArray(arr, n, it+1);
    }
}

int main()
{
    int arr[10] = {6, 9, 8, 4, 5, 1, 2, 3, 4, 5};
    int n = 10;
    printArray(arr, n, 0);
    return 0;
}
```

9. Find the largest element of a given array.

```
#include<iostream>
using namespace std;
int largestElement(int arr[], int currentIndex)
{
    if(currentIndex==0)
        return arr[currentIndex];
    else
    {
        int currentValue = arr[currentIndex];
        int friend_ = largestElement(arr, currentIndex-1);
        return max(currentValue, friend_);
    }
}
int main()
{

    int arr[10] = {6, 9, 8, 4, 5, 1, 2, 3, 4, 5};
    int n = 10;
    int result = largestElement(arr, n-1);
    cout << result << endl;
    return 0;
}
```

10. Find the smallest element of a given array.

```
#include<iostream>
using namespace std;
int smallestElement(int arr[], int currentIndex)
{
    if(currentIndex==0)
        return arr[currentIndex];
    else
    {
        int currentValue = arr[currentIndex];
        int friend_ = smallestElement(arr, currentIndex-1);
        return min(currentValue, friend_);
    }
}
int main()
{

    int arr[10] = {6, 9, 8, 4, 5, 1, 2, 3, 4, 5};
    int n = 10;
```

```cpp
    int result = smallestElement(arr, n-1);
    cout << result << endl;
    return 0;
}
```

11. Find the largest and smallest element of a given array.

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

struct MinMax
{
    int minimum;
    int maximum;
};

MinMax minMaxElement(int arr[], int currentIndex)
{
    if(currentIndex==0)
        return {arr[currentIndex], arr[currentIndex]};
    else
    {
        int currentValue = arr[currentIndex];
        MinMax friend_ = minMaxElement(arr, currentIndex-1);

        MinMax result = friend_;

        if(currentValue<result.minimum)
            result.minimum = currentValue;
        else if(currentValue>result.maximum)
            result.maximum = currentValue;

        return result;
    }
}

int main()
{

    int arr[10] = {6, 9, 8, 4, 5, 1, 2, 3, 4, 5};
    int n = 10;
    MinMax result = minMaxElement(arr, n-1);
    cout << result.minimum << " " << result.maximum << endl;
    return 0;
}
```

12. Check whether a given string is palindrome or not.

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

bool isPalindrome(string str, int left, int right)
{
    if(left>=right)
        return true;
    else
    {
        if(str[left]==str[right])
        {
            bool friend_ = isPalindrome(str, left+1, right-1);
            return friend_;
        }
        else
            return false;
    }
}
int main()
{
    string str = "madam";

    bool result = isPalindrome(str, 0, str.size() - 1);

    if(result==true)
        cout << "Palindrome\n";
    else
        cout << "Not a Palindrome\n";

    return 0;
}
```

13. Remove white spaces from a string & convert upper cases to lower.

```cpp
#include<iostream>
using namespace std;
string removeWhiteSpaceCaseConversion(string str, int it)
{
    if(it==str.size()) return "";
    else
    {
        char currentCharacter = str[it];

        string friend_ = removeWhiteSpaceCaseConversion(str, it+1);


        if(currentCharacter>='A' && currentCharacter<='Z')
        {
            currentCharacter = currentCharacter - 'A';
            currentCharacter = currentCharacter + 'a';

        }


        return currentCharacter == ' ' ? friend_ : currentCharacter+friend_;
    }

}

int main()
{
    string str;
    getline(cin, str);
    string result = removeWhiteSpaceCaseConversion(str, 0);
    cout << result << endl;
    return 0;
}
```