

# CSE111: Data Structures

## Assignment - Recursion

Name: **Mahros Mohamed Mahros alqabasy**

ID: **223106831**

Program: **CS**

(1)

What is the output of the following program?

```
#include <iostream>
using std::cout;
using std::endl;

int mystery(int n);
//Precondition n >= 1.

int main( )
{
    cout << mystery(3) << endl;
    return 0;
}

int mystery(int n)
{
    if (n <= 1)
        return 1;
    else
        return ( mystery(n - 1) + n );
}
```

- Mystery function will return the total sum from n to 1
- $\text{Sum}(3) = 3 + 2 + 1 = 6$

What is the output of the following program? What well-known mathematical function is rose?

```
#include <iostream>
using std::cout;
using std::endl;

int rose(int n);
//Precondition: n >= 0.

int main( )
{
    cout << rose(4) << endl;
    return 0;
}

int rose(int n)
{
    if (n <= 0)
        return 1;
    else
        return ( rose(n - 1) * n );
}
```

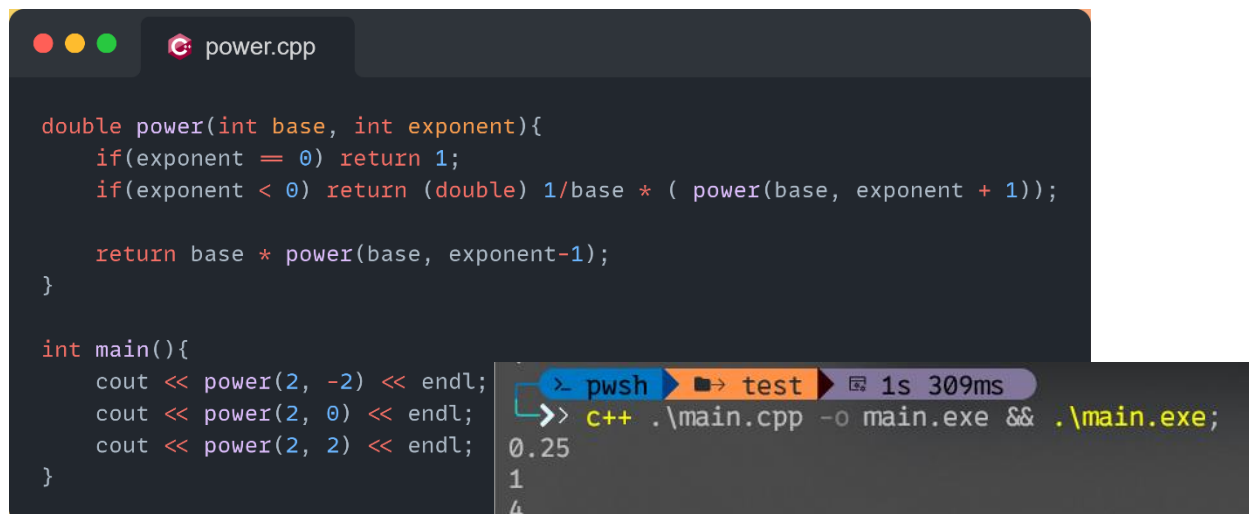
- This program will calculate the factorial of the given number
- Factorial(4) =  $4 \times 3 \times 2 \times 1 = 24$

(3)

Redefine the function power so that it also works for negative exponents. In order to do this, you will also have to change the type of the value returned to double. The function declaration and header comment for the redefined version of power are as follows:

```
double power(int x, int n);
//Precondition: If n < 0, then x is not 0.
//Returns x to the power n.
```

*Hint:  $x^{-n}$  is equal to  $1/(x^n)$ .*



```
power.cpp

double power(int base, int exponent){
    if(exponent == 0) return 1;
    if(exponent < 0) return (double) 1/base * ( power(base, exponent + 1));

    return base * power(base, exponent-1);
}

int main(){
    cout << power(2, -2) << endl;
    cout << power(2, 0) << endl;
    cout << power(2, 2) << endl;
}
```

pwsh test 1s 309ms  
c++ .\main.cpp -o main.exe && .\main.exe;  
0.25  
1  
4

(4)

Write a recursive function that has an argument that is an array of characters and two arguments that are array indexes. The function should reverse the order of those entries in the array whose indexes are between the two bounds. For example, if the array is

```
a[1] == 'A' a[2] == 'B' a[3] == 'C' a[4] == 'D' a[5] == 'E'
```

and the bounds are 2 and 5, then after the function is run, the array elements should be

```
a[1] == 'A' a[2] == 'E' a[3] == 'D' a[4] == 'C' a[5] == 'B'
```

Embed the function in a program and test it. After you have fully debugged this function, define another function that takes a single argument that is an array that contains a string value; the function should reverse the spelling of the string value in the array argument. This function will include a call to the recursive definition you did for the first part of this project. Embed this second function in a program and test it.

(5)

Write a recursive function named `contains` with the following header:

```
bool contains (char *haystack, char *needle)
```

The function should return `true` if the C-string `needle` is contained within the C-string `haystack` and `false` if `needle` is not in `haystack`. For example,

```
contains("C++ programming", "ogra") should return true
```

```
contains("C++ programming", "grammy") should return false
```

You are not allowed to use the `string` class `substr` or `find` functions to determine a match.

(6)

What is printed from *fun*('G') as coded below? Explain what the function does.

```
void fun (char c)
{
    if ( c < 'A' || c > 'Z')
    {
        return;
    }
    cout << c << " ";
    fun (c + 1);
}
```

This will print only if given char is alpha. Then print from given char to Z

$F(G) = G H I J K L M N O P Q R S T U V W X Y Z.$

(7)

What is returned from *fun*(5, 12) and *fun*(12, 5) as coded below? Explain what the function does.

```
int fun (int n, int m)
{
    if ( n == m)
    {
        return 0;
    }
    else if ( n > m)
    {
        return 1;
    }
    else
    {
        return fun (m, n);
    }
}
```

$F(5, 12) = 1$

$F(12, 5) = 1$

(8)

What is printed from *fun*(5) as coded below? Explain what the function does.

```
void fun (int n)
{
    if (n < 0)
    {
        return;
    }
    cout << n << " ";
    fun (n - 1);
}
```

Prints numbers from n to 0

$F(5) = 5 4 3 2 1 0$

(9)

What is returned from  $fun(5)$  as coded below? Explain what the function does.

```
double fun (int n)
{
    if (n <= 1)
    {
        return 0;
    }
    return fun (n - 1) + 1.0 / n;
}
```

$$f(n) \rightarrow \sum_{k=1}^{n-1} k^{-1}$$

$f(5)$ :

$$\frac{1}{5} + f(4)$$

$$\frac{1}{4} + f(3)$$

$$\frac{1}{3} + f(2)$$

$$\frac{1}{2} + f(1)$$

$$result \rightarrow \frac{1}{5} + \frac{1}{4} + \frac{1}{3} + \frac{1}{2} + 0$$

(10)

What is returned from  $fun(4)$  as coded below? Explain what the function does.

```
double fun (int n)
{
    if (n <= 1)
    {
        return 0;
    }
    return fun (n - 1) + (double) n / (n + 2);
}
```

Really, I do not know any way to explain this fun like mathematical rule of summation.

$$f(n) \rightarrow \sum_{k=1}^{n-1} \frac{n}{n+2}$$

$f(4)$ :

$$\frac{4}{6} + f(3)$$

$$\frac{3}{5} + f(2)$$

$$\frac{2}{4} + f(1)$$

$$result \rightarrow \frac{4}{6} + \frac{3}{5} + \frac{2}{4} + 0$$

(11)

What is returned from  $fun(4)$  as coded below? Explain what the function does.

```
double fun (int n)
{
    if (n <= 1)
    {
        return 0;
    }
    return fun (n - 1) + double (n) / (2 * n - 1);
}
```

$$f(n) \rightarrow \sum_{k=1}^{n-1} \frac{n}{2n-1}$$

$f(4)$ :

$$\frac{4}{7} + f(3)$$

$$\frac{3}{5} + f(2)$$

$$\frac{2}{3} + f(1)$$

$$result \rightarrow \frac{4}{7} + \frac{3}{5} + \frac{2}{3} + 0$$

(12)

What is returned from  $fun(4)$  as coded below? Explain what the function does.

```
double fun (int n)
{
    if (n <= 1)
    {
        return 0;
    }
    return fun (n - 1) + double (n) / (3 * n - 1);
}
```

$$f(n) \rightarrow \sum_{k=1}^{n-1} \frac{n}{3n-1}$$

$f(4)$ :

$$\frac{4}{11} + f(3)$$

$$\frac{3}{8} + f(2)$$

$$\frac{2}{5} + f(1)$$

$$result \rightarrow \frac{4}{11} + \frac{3}{8} + \frac{2}{5} + 0$$

(13)

What is printed from *fun*(164) as coded below? Explain what the function does.

```
int fun (int n)
{
    if (n < 10)
    {
        return n
    }
    else
    {
        cout << (n % 10);
        return fun (n / 10);
    }
}
```

*this will reverse digits in given number*

***f(164):***

***4 → f(16)***

***6 → f(1)***

***end the loop***

Result 46

This function has logical error: when we enter 164 it will print '46' but it must be '461' \* to solve this we have to print numbers inside first condition

```
if(n < 10){
    cout << n;
    return n;
}
```

I don't know what is the business for this function but it's not required to return any value.

(14)

What is printed from *fun*(19) as coded below? Explain what the function does.

```
void fun (int n)
{
    if (n > 0)
    {
        fun (n / 2);
        cout << n % 2;
    }
}
```

*this will print reminder for every  $n \% 2$*

***f(19)***

***f(9) → 0.5 → 0***

***f(4) →***

## 15. Recursive Power Function

Write a function that uses recursion to raise a number to a power. The function should accept two arguments: the number to be raised and the exponent. Assume that the exponent is a nonnegative integer. Demonstrate the function in a program.

```
double power(int base, int exponent){
    if(exponent == 0) return 1;
    if(exponent < 0) return (double) 1/base * ( power(base, exponent + 1));

    return base * power(base, exponent-1);
}

int main(){
    cout << power(2, -2) << endl;
    cout << power(2, 0) << endl;
    cout << power(2, 2) << endl;
}
```





```
> pwsh test 3ms
>> c++ .\main.cpp -o main.exe && .\main.exe;
Simple power program.
accepts two inputs:
base: int, exponent:int.
return long double.

1 0
Power(1, 0) ==> 1.00000
1 2
Power(1, 2) ==> 1.00000
2 10
Power(2, 10) ==> 1024.00000
3 50
Power(3, 50) ==> 717897987691852578422784.00000
100 -2
Power(100, -2) ==> 0.00010
10 -1
Power(10, -1) ==> 0.10000
50 -3
Power(50, -3) ==> 0.00001
0 0
```

## 16. Sum of Numbers

Write a function that accepts an integer argument and returns the sum of all the integers from 1 up to the number passed as an argument. For example, if 50 is passed as an argument, the function will return the sum of 1, 2, 3, 4, ... 50. Use recursion to calculate the sum. Demonstrate the function in a program.



power.cpp

```
int sum(int n){
    if(n == 1) return 1;
    return n + sum(n-1);
}

int main(){
    printf("Simple sum program.\n accepts one input:\n n: int \nreturn int.\n\n");
    int n;
    while (cin >> n && n != 0){
        printf("Summation(%i = %i)\n", n, sum(n));
    }
}
```



```
>_ pwsh test 1ms
>> c++ .\main.cpp -o main.exe && .\main.exe;
Simple sum program.
 accepts one input:
 n: int
return int.

1
Summation(1 = 1)
10
Summation(10 = 55)
20
Summation(20 = 210)
250
Summation(250 = 31375)
360
Summation(360 = 64980)
10360
Summation(10360 = 53669980)
```

What is the output of the following programs?

```

#include <iostream>
using namespace std;

int function(int);

int main()
{
    int x = 10;

    cout << function(x) << endl;
    return 0;
}

int function(int num)
{
    if (num <= 0)
        return 0;
    else
        return function(num - 1) + num;
}

```

$$f(n) = \sum_{k=1}^n k$$

*output*  $\rightarrow 10 + 9 + 8 + 7 + 6$   
 $\quad \quad + 5 + 4 + 3 + 2 + 1$   
 $\quad \quad + 0 \rightarrow 55$

20. What is the output of the following programs?

```

#include <iostream>
using namespace std;

void function(int);

int main()
{
    int x = 10;

    function(x);
    return 0;
}

void function(int num)
{
    if (num > 0)
    {
        for (int x = 0; x < num; x++)
            cout << '*';
        cout << endl;
        function(num - 1);
    }
}

```

*this will print trigometric*

```

*****
*****
*****
*****
*****
*****
*****
****
***
**
*

```

20. What is the output of the following programs?

```

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

void function(string, int, int);

int main()
{
    string mystr = "Hello";
    cout << mystr << endl;
    function(mystr, 0, mystr.size());
    return 0;
}

void function(string str, int pos, int size)
{
    if (pos < size)
    {
        function(str, pos + 1, size);
        cout << str[pos];
    }
}

```

Print Hello reversed

Like this:  
olleH