UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE INGENIERIA DE PRODUCCION Y SERVICIOS

ESCUELA PROFESIONAL DE CIENCIAS DE LA COMPUTACIÓN

LABORATORIO 18 – Meta programación.

DOCENTE:

Enzo Edir Velásquez Lobatón

ALUMNO:

Owen Haziel Roque Sosa.

FECHA:

04/08/2022

Arequipa – Perú

1. Suma los dígitos de un numero de forma recursiva utilizando meta programación.

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

template <int N>
struct SumDigitos {
    enum {
        value = (N % 10) + SumDigitos<N/10>::value
    };
};
template <>
struct SumDigitos <0> {
    enum { value = 0 };
};

int main(int argc, char *argv[]) {
    int x = SumDigitos<1564>::value;
    cout << "Suma de 1564 -> " << x << endl;
    system("pause");
    return 0;
}
```

```
Suma de 1564 -> 16
Presione una tecla para continuar . . .
```

2. Calcular el valor de la posición Fibonacci usando recursividad utilizando meta programación.

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

template <unsigned int N>
struct posFibonacci {
    enum {
        value = posFibonacci<N-1>::value +
                posFibonacci<N-2>::value
    };
};

template <>
struct posFibonacci <0> {
    enum { value = 0 };
};

template <>
struct posFibonacci <1> {
    enum { value = 1 };
};

int main(int argc, char *argv[]) {
    int x = posFibonacci<5>::value;
    cout << "posFibonacci<5> -> " << x << endl;
    system("pause");
    return 0;
}
```

```
posFibonacci<5> -> 5
Presione una tecla para continuar . . .
```

3. Calcula la potencia de un numero de forma recursiva utilizando meta programación

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

template <int B, int EXP>
struct Power
{
    enum { value = B * Power<B, EXP-1>::value };
};
template <int B>
struct Power <B, 0>
{
    enum { value = 1 };
};

int main(int argc, char *argv[]) {
    int x = Power<0,6>::value;
    cout << "Potencia de 4 a la 6 -> " << x << endl;
    system("pause");
    return 0;
}
```

```
Potencia de 4 a la 6 -> 4096
Presione una tecla para continuar . . .
```

4. Construya una función recursiva que convierta un número decimal en una cadena que represente el valor del número en hexadecimal (base 16) utilizando meta programación.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

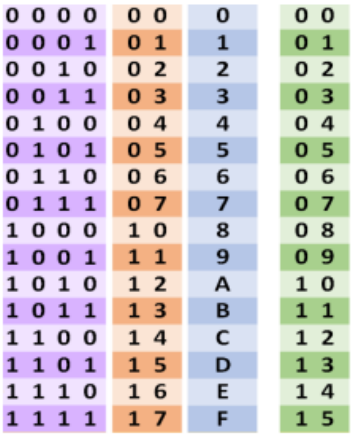
```
#include <iostream>
using namespace std;

string res;
template <int N>
void hexa() {
    hexa<N/16>();
    if(N%16>9){
        //cout<<char(N%16+55);
        res.push_back(char(N%16+55));
    }
    else {
        //cout<<N%16;
        res.append(to_string(N%16));
    }
}
template<>
void hexa<0>(){
    return;
}

int main(int argc, char *argv[]) {
    hexa<1535>();
    cout << "1535 a Hex -> " << res << endl;
    system("pause");
    return 0;
}
```

```
1535 a Hex -> 5FF
Presione una tecla para continuar . . .
```

5. Ingresar un número y mostrar su equivalente en binario usando una función recursiva utilizando meta programación.



0 0 0 0	0 0	0	0 0
0 0 0 1	0 1	1	0 1
0 0 1 0	0 2	2	0 2
0 0 1 1	0 3	3	0 3
0 1 0 0	0 4	4	0 4
0 1 0 1	0 5	5	0 5
0 1 1 0	0 6	6	0 6
0 1 1 1	0 7	7	0 7
1 0 0 0	1 0	8	0 8
1 0 0 1	1 1	9	0 9
1 0 1 0	1 2	A	1 0
1 0 1 1	1 3	B	1 1
1 1 0 0	1 4	C	1 2
1 1 0 1	1 5	D	1 3
1 1 1 0	1 6	E	1 4
1 1 1 1	1 7	F	1 5

```cpp
#include <iostream>
using namespace std;
template <int N>
void toBinary() {
    toBinary<N/2>();
    cout << N%2;
}
template <>
void toBinary<0>() {
    return;
}

int main(int argc, char *argv[]) {
    cout << "Binario de 18 -> ";
    toBinary<18>();
    return 0;
}
```

```
Binario de 18 -> 10010
```