Ejercicio 1: Fibonacci con Programación Dinámica

```
#include<iostream>
//tamaño N matriz
#define N 100
using namespace std;
//inicializar valores con -1
const int nd = -1;
int memo[N];

void inicializar()
{
	for (int i = 0; i < N; i++)
		memo[i] = nd;
}

int fibonacci_memo(int n) {
	//Si memo[n] es igual a -1, ingresar significa que no esta
almacenado;
	//no ingresa a la condición muestra el resultado
	if (memo[n] == nd)
	{
		if (n <= 1)//si es el primer elemento
			memo[n] = n;
		else //almacena en memo el resultado
			memo[n] = fibonacci_memo(n - 1) + fibonacci_memo(n -
2);
	}
	return memo[n];
}

int main() {
//	inicializar();
	memset(memo, -1, sizeof memo); //coloca todos los valores en -1
	cout << fibonacci_memo(6);
	return 0;
}
```

****

Ejercicio 2: Ejercicios números feos, significa son divisibles 2, 3, 5
//1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, ...

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

```
int maxDivide(int a, int b) {
	while (a % b == 0) //9%3 =0, 3%3 = 0
		a = a / b; // 9/3 = 3, 3/3 =1
	return a;
}

//Verificar si es el número es feo

int isFeo(int no) {
	no = maxDivide(no, 2);
	no = maxDivide(no, 3);
	no = maxDivide(no, 5);
	return (no == 1) ? 1 : 0;
}

int obtenerFeoNo(int n) {
	int i = 1;
	int count = 1;
	while (n > count) {
		i++;
		//si el resultado 1 permite contar
		if (isFeo(i))
			count++;
	}
	return i;
}

int main() {
	int no = obtenerFeoNo(150);
	cout << "\nEl valor es: " << no;
	cin.get();
	cin.ignore();
	return 0;
}


//with Dinamic Programming


#include<iostream>
using namespace std;

const int n = 7;

//(1) 1×2, 2×2, 3×2, 4×2, 5×2, …
//(2) 1×3, 2×3, 3×3, 4×3, 5×3, …
```

```
//(3) 1×5, 2×5, 3×5, 4×5, 5×5, …

int obtenerFeoN() {

	int Feo[n];
	int i2 = 0, i3 = 0, i5 = 0;
	int multiple_2 = 2;
	int multiple_3 = 3;
	int multiple_5 = 5;
	int sig_feo_n = 1;
	Feo[0] = 1;

	for (int i = 1; i < n; i++) {
		sig_feo_n = min(multiple_2, min(multiple_3, multiple_5));
		Feo[i] = sig_feo_n;
		if (sig_feo_n == multiple_2)
		{
			i2 = i2 + 1;
			multiple_2 = Feo[i2] * 2;
		}
		if (sig_feo_n == multiple_3) {
			i3 = i3 + 1;
			multiple_3 = Feo[i3] * 3;
		}
		if (sig_feo_n == multiple_5) {
			i5 = i5 + 1;
			multiple_5 = Feo[i5] * 5;
		}
	}
	return sig_feo_n;
}

int main() {
	cout << obtenerFeoN();
	cin.get();
	cin.ignore();
	return 0;
}



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

Ejercicio 3: Cuenta cantidad de monedas

#include<iostream>
```

```
using namespace std;
const int nd = -1;
const int N = 3;
const int tam = 1001;

int monedas[N] = { 1,2,3 };
int memo[1001];

void inicializar()
{
	for (int i = 0; i < 1000; i++)
		memo[i] = nd;
}

int cambio(int valor) {
	if (valor < 0) return tam;
	if (valor == 0) return 0;
	if (memo[valor] != nd) return memo[valor];
	//ingresa un valor alto, para en la primera preguntar por el mínimo
y elija otro número
	int res=tam;
	//obtenemos el mínimo valor de la secuencia de valores del 1, 2 , 3
		for (int i = 0; i < N; i++) //obtiene el mínimo
valor de ambos.
			res = min(res, cambio(valor - monedas[i]));
		return memo[valor]=res+1;	//como son monedas se
aumenta en uno
}

int main() {
	inicializar();
	//memset(memo, -1, sizeof memo); //coloca todos los valores -1

	int val;
	cin >> val;
	cout << "Resultado: " << cambio(val)<<endl<<endl;
	int valor = val;
	for (int i = 0; i <= valor; i++)
		cout << i << " = " << cambio(i) << endl;
	cin.get();
	cin.ignore();
	return 0;
}
```