* Código más corto.
* En algunos casos consume más memoria y tiempo.
* El **caso Base es** la condición de corte. En el caso base no se llama nunca a la función.
* Un problema donde cada vez se haga más chico.
* Como mínimo tiene q haber un caso base.
* No se utilizan variables auxiliares´.
* **La función se llama a si misma hasta que se cumpla el caso base.**
* Todos los resultados que hayan invocado a la función antes, se suman cando se cumple el caso base. Cuando se guarda el llamado a la función en una variable, este se va guardando y luego la llamamos para que ejecute el resultado que conserva adentro.
* Cada resultado se guarda en un FRAME para después ser sumados.
* FRAME = Segmento de Memoria.
* Siempre que este la función recursiva dentro de otra función, hay que invocarla con el RETURN dentro de la FUNCION PRINCIPAL.
* Las cosas se inicializan fuera de la función recursiva
* Cuando un programa comienza su ejecución, se crea la PILA DE ACTIVACION, y cada elemento que se agregue a la pila ocupara un nuevo segmento de memoria.
* Cada llamado a la función crea una nueva entrada **en la pila de activación**
* La **técnica de recursividad** se basa en una función que se llama a si misma hasta llegar a un caso base que es quien cortara la recursividad. La función, al llamarse a sí misma, carga los datos e instrucciones en la pila de activación. Este proceso termina cuando se llega al caso base (condición de corte), empezando a Desapilar lo almacenado o cuando la memoria de la pila se llena por completo, arrojando un error tipo “Stack overflow” Pueden existir varios casos bases. Para aplicarla esta técnica el problema debe ser siempre el mismo pero “más pequeño” en cada invocación para acercarse al él o los casos bases