Resolucion

DUDAS: Ejercicio 2

Ejercicio 1)

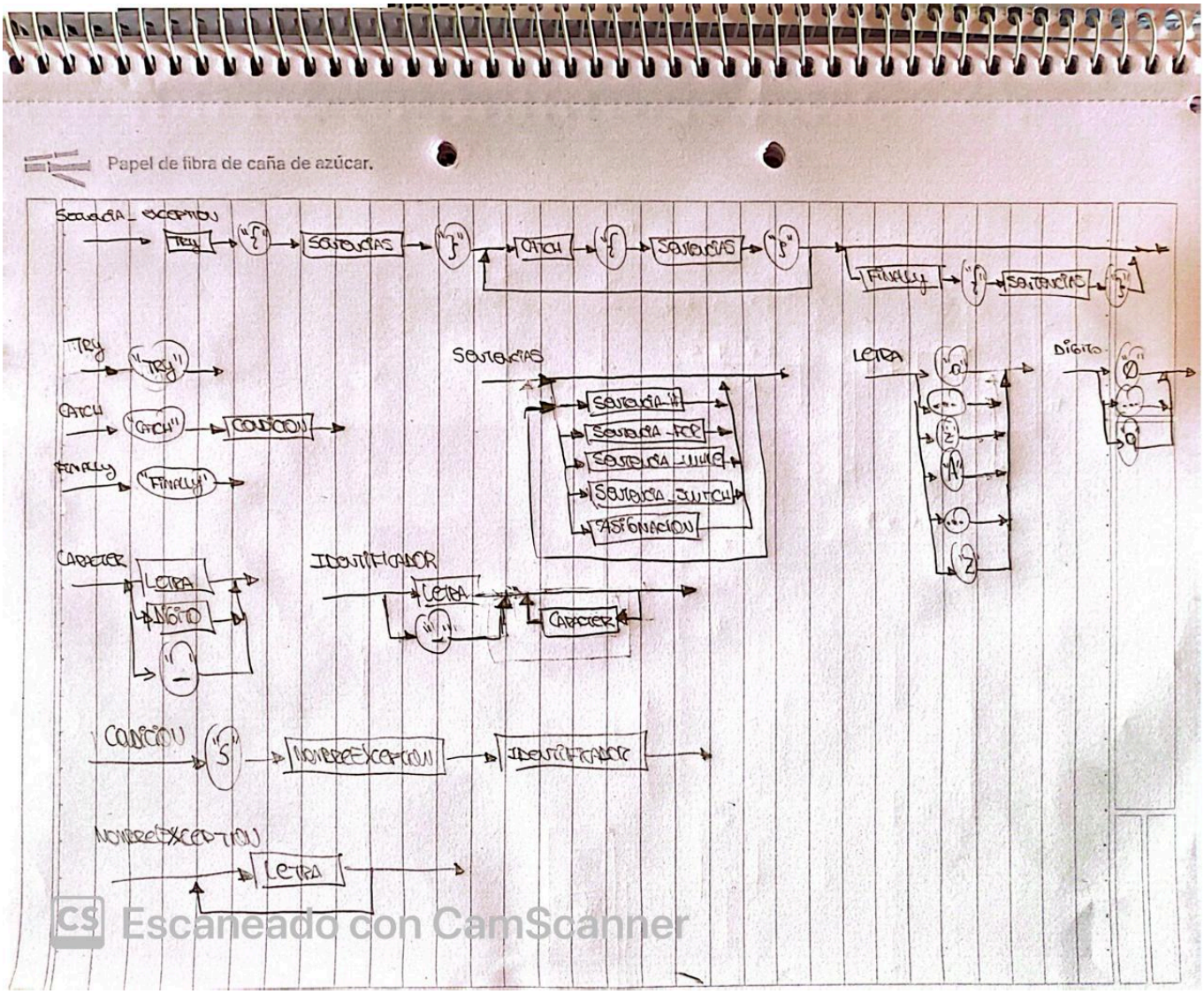
Inciso A)

EBNF para bloque de manejo de excepciones en java.
Tener en cuenta que java admite try/catch anidados.

```
G = ( N , T , S , P )

N = <sentencia_exception>, <catch>, <try>, <letra>, <digito>, <caracter>,
<condicion>, <identificador>, <asignacion>, <nombreException>, <sentencia>,
<sentencia_if>, <sentencia_for> , <sentencia_while>, <sentencia_switch>

T = "try", "catch", "finally", " = " "_", "(" ")", "{" , "}",
0,1,2,3,4,5,6,7,8,9
"a","b"...,"z", "A","B", ... "Z"

S = <sentencia_exception>

P = {

<sentencia_exception>::=  <try> "{" <sentencias> "}"
                          { <catch> "{" <sentencias> "}" }+ //"+"por catch
anidados
                          [ <finally> "{" <sentencias> "}" ]

<try> ::= "try"
<catch> ::= "catch" <condicion>
<finally> ::= "finally"
<letra>::= "a" | "b" | ...| "z" | "A" | "B" |... | "Z"
<caracter> ::= ( <letra> | <digito> | "_" )
<digito> ::= 0|1|2|3|4|5|6|7|8|9

<condicion>::= " ( " <nombreException> <identificador> " ) "
// por ejemplo ( Exception ex )

<nombreException> ::= {letra}+
<identificador> ::=  ( <letra> {caracter}*  | "_" {caracter}* )
```

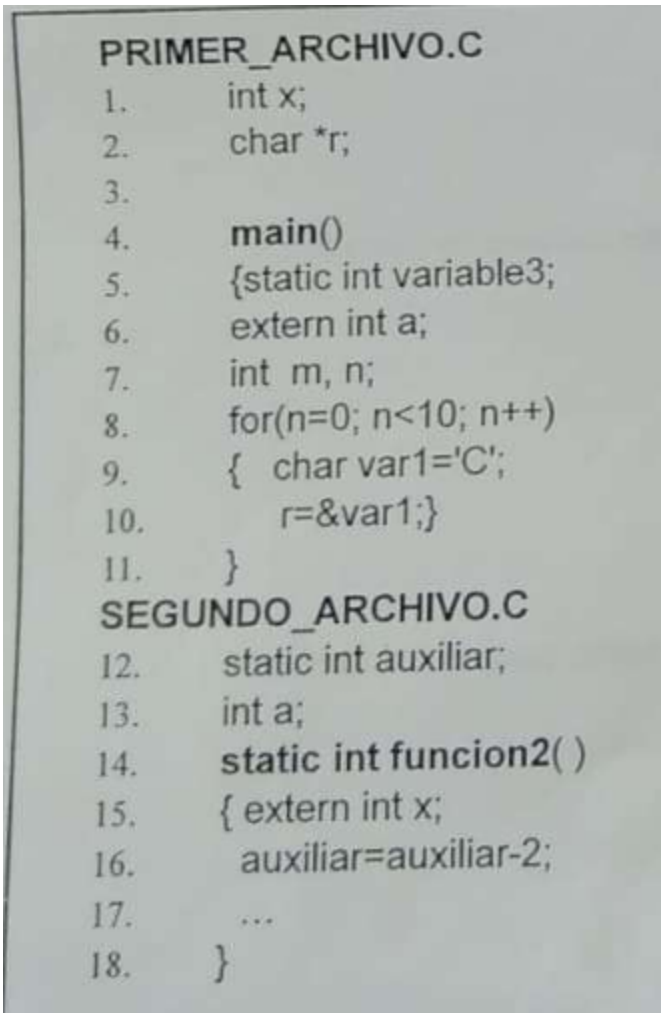
```
<sentencias> ::= { (
                <sentencia_if> |
                <sentencia_for> |
                <sentencia_while> |
                <sentencia_switch> |
                <sentencia_exceptiom> |
                <asignacion> |
                ....otras sentencias....
                ) }*

<asignacion> ::= <identificador> " = " <valor>
<valor>::= ( <identificador> | {digito}+)
<sentencia_if> ::= .....
<sentencia_for> ::= ....
<sentencia_while> ::= ....
<sentencia_switch> ::= ....

}
```

Inciso B)

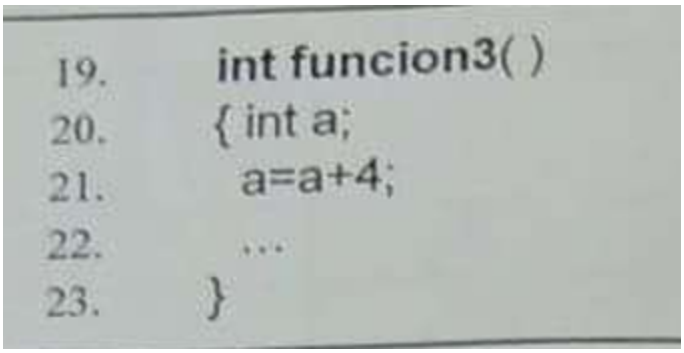


Ejercicio 2

Consultar resolucion



```
PRIMER_ARCHIVO.C
1.      int x;
2.      char *r;
3.
4.      main()
5.      {static int variable3;
6.      extern int a;
7.      int  m, n;
8.      for(n=0; n<10; n++)
9.      {  char var1='C';
10.        r=&var1;}
11.     }
SEGUNDO_ARCHIVO.C
12.     static int auxiliar;
13.     int a;
14.     static int funcion2( )
15.     { extern int x;
16.       auxiliar=auxiliar-2;
17.       ...
18.     }
```



```
19.     int funcion3( )
20.     { int a;
21.       a=a+4;
22.       ...
23.     }
```

Consideraciones a tener en cuenta:

- Las variables globales se inicializan con R-Valor 0
  - Tienen tiempo de vida desde el momento 1 hasta que termina todo el codigo
  -
- Las variables estaticas se inicializan con R-Valor 0
  - Se alocan al momento de alocar el codigo. Es decir, su tiempo de vida seria desde el instante 1 hasta el ultimo instante del codigo
- Las variables locales son automaticas y tienen R-Value basura.
- Las funciones (funciones estaticas y funciones no estaticas) se comportan de la misma forma ?

- Su alcance : Es desde una linea siguiente a su declaracion, hasta que finaliza el bloque de la funcion? o hasta que finaliza el bloque que lo contiene?
- Su tiempo de vida: es desde que se declara la funcion, hasta que termina su declaracion?

Identificacor	L - Valor	R - Value	Alcance	T.V
x linea 1	automatica	0	2..11 16..18 ??	1..23
r linea 2	automatica	null	3. 11	1..11? o 1..23?
* r linea 2	dinamica	bsaura	3..10	10
main linea 4	-	-	5..11	4..11
variable3 linea 5	estatica	0	6 ..11	<1..23>
m linea 7	automatica	basura	8..11	4..11
n linea 7	automatica	basura	8..11	4.11
var1 linea 9	automatica	C ? o basura?	10	8..10
auxiliar linea 12	estatica	0	13..23	<1..23>
a linea 13	automatica	0	7..11 14..23	1..23
fun2 ( ) linea 14	-	-	15..23	1..23? por ser estatica? 0 14..18 ?
int x linea 15	automatica	basura	16..18	14..18
fun3 ( )	-	-	20..23	19..23
a linea 20	automatica	basura	21..23	19..23

Ejercicio 3

a) Todos los lenguajes funcionales son fuertemente tipados

FALSO

No todos los lenguajes funcionales son fuertemente tipados.
Por ejemplo python permite programar con estilo funcional pero no es fuertemente tipado, python tiene tipado dinamico.

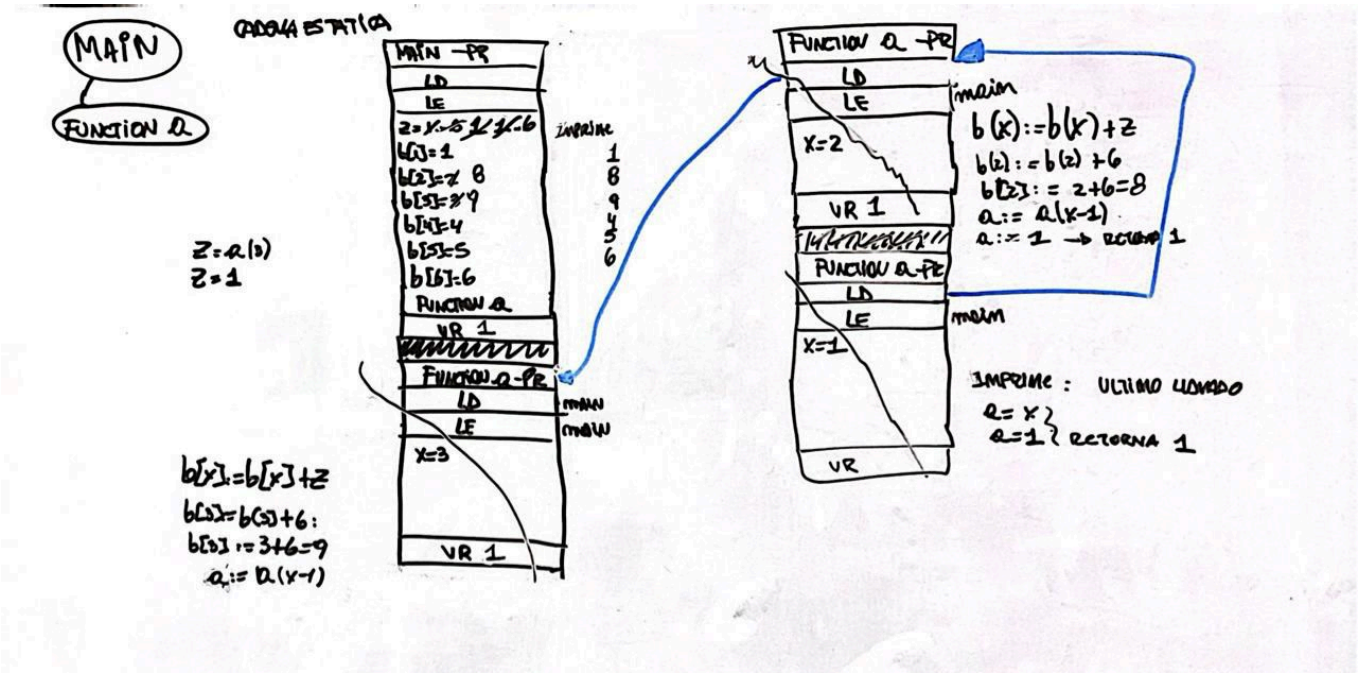
b) Los parametros formales por resultado pueden usarse en ejecucion tal como se reciben en el procedimiento o funcion?

FALSO

Los parametros por resultado (out) no tienen un valor valido al comienzo de la ejecucion del procedimiento o funcion. Para usar correctamente este tipo de parametos, se le debe asignar un valor dentro del procedimiento o funcion, y ese valor será devuelto. Es decir, el procedimiento o funcion debe estar obligado a asignarle un valor al parametro por resultado.

Ejercicio 4

En esta cursada (hasta el primer parcial) no vimos pila de ejecucion con parametros, pero creo que llegue al resultado correcto, en el compilador se llego a lo mismo. (los compiladores manejan cadena estatica.)



```
program queImprime;
var
z : integer;
b: array [1..6] of integer;
function a (x : integer) : integer;
begin
   if (x = 1) then begin
      writeln ('Ultimo llamado ');
      a:= x;
   end
   else begin
      b[x]:= b[x] + z;
      a:= a(x-1);
   end;
end;

begin
   for z:= 1 to 6 do begin
      b[z] := z ;
   end;
   z := a(3);
   writeln ('El valor de Z al final de todo es: ',z);

   writeln ('Imprimiendo el vector: ');
   for z:= 1 to 6 do
    write (b[z], ' ' );
end.
```

```
C:\WINDOWS\SYSTEM32\cmd.exe
Ultimo llamado
El valor de Z al final de todo es: 1
Imprimiendo el vector:
1 8 9 4 5 6

-----------------
(program exited with code: 0)

Presione una tecla para continuar . . .
```

Los demas incisos que no los hice es por que en mi cursada no los vemos para el primer parcial. Son contenidos que los modificaron y los toman en el segundo parcial.