Pruebas.

Nota: Las celdas con fondo azul requirieron corrección, las verdes no, las rojas fallan y las que están en blanco no hay información al respecto.

Caso de prueba	Esperado	Resultado

#1
const int pi = 456;
int def main() {
return 0;
}
	Compilación exitosa	Compilación exitosa

#2
const int pi = 456;
const int pi2 = 456;
const int pi3 = 456;
int def main() {
return 0;
}
	Compilación exitosa	Compilación exitosa

#3
const int pi = 456;
const int pi2 = 456
const int pi3 = 456;
int def main() {
return 0;
}
	Was expected semicolon.	Was expected semicolon.

#4
int def main() {
int a = 5.1;
int x = 3;
return 0;
}
	Compilación exitosa	Successful compilation

#5
int def main() {
int a = 0, x, h = 3;
int b;
return 0;
}
	Compilación exitosa	Compilación exitosa

#6 int def main() { chr x = ‘a’, b = ‘a’;	Compilación exitosa	Compilación exitosa

return 0; }		

#7
int def main() {
str david = “hola mundo”;
str onder;
str miguel = “hola ”, dons = “mundo”;
return 0;
}
	Compilación exitosa	Compilación exitosa

#8
int def main() {
bool x = true;
bool t = x, r = false;
return 0;
}
	Compilación exitosa	Compilación exitosa

#9
int def main() {
dec aba = 2.1;
dec x = 0, y, z = aba;
dec yr = 45;
return 0;
}
	Compilación exitosa	Compilación exitosa

#10
int def main() {
int uno = 1;
dec aba = 2.1, abo;
bool x;
chr letra = 'l';
chr letra2 = '2', letra3 = '3';
str cadena = "hola mundo!";
return 0;
}
	Compilación exitosa	Compilación exitosa

#11
int def main() {
int x = 2;
x += 4+8;
int a = 2;
x = sumar(x, a);
return 0;
}
	Compilación exitosa	Compilación exitosa

#12 int def main() {	Compilación exitosa	Compilación exitosa

int x = 2;
x += 4+8;
int a = 2, b = 2;
x = sumar(a, b);
a = restar(5, x, true);
imprimirResultado(x);
return 0;
}		

#13
int def main() {
int x = 2;
int a;
x += 4+8;
x = 4*3*a+45;
return 0;
}
	Compilación exitosa	Compilación exitosa

#14
int def main() {
int x = 2;
int a;
x *=5; // potencia
a **= 5%x+3; // compuesta
return 0;
}
	Compilación exitosa	Compilación exitosa

#15
int def main() {
int x = 2;
int a;
a _/= 5%x+3; // compuesta
return 0;
}
	Compilación exitosa	Compilación exitosa

#16
int def main() {
int x = 2;
int a;
llamadaAFuncion();
return 0;
}
	Compilación exitosa	Compilación exitosa

#17
int def main() {
int x = 5;
if(a == 5){
	Compilación exitosa	Compilación exitosa

} return 0; }		

#18
int def main() {
int x = 5;
int a, o;
if(a != 5 AND a != x)
return 0;
}
	Compilación exitosa	Compilación exitosa

#19
int def main() {
int x = 5;
int a, o;
if(a != b AND x > 5 OR a < 10 XOR a >= 14) {
 	
}
return 0;
}
	Compilación exitosa	Compilación exitosa

#20
int def main() {
int x = 5;
int a, o;
if(a == true){}
else if(a == 1){}
else {}
return 0;
}
	Compilación exitosa	Compilación exitosa

#21
int def main() {
int x = 5;
int a = 0, b = 3;
if(a != b AND b == 8){
 	
}
else if(a <= 1){
 	
}
return 0;
}
	Compilación exitosa	Compilación exitosa

#22 int def main() {	Compilación exitosa	Compilación exitosa

int x = 5;
int a = 0, b = 3;
if(1 == 1){
 	
}
else {
 	
}
return 0;
}		

#23
int def main() {
int x = 5;
int a = 0, b = 3;
if(a == 1){
int x = 2;
}
return 0;
}
	Compilación exitosa	Compilación exitosa

#24
int def main() {
int x = 5;
int a = 0, b = 3;
if(a == 1){
int x = 2;
if(x == 2){
x += 1;
}
else {
x -= 2;
if(x < 0) {
x = 0;
}
else if(x == 1) {
unaFuncion();
}
else {
str cadena = "hola mundo";
otraFuncion(cadena);
}
}
}
return 0;
}
	Compilación exitosa	Compilación exitosa

#25
int def main() {
int x = 5;
int a = 0, b = 3;
if(a == 1){
if(a == 2) {
if(a == 3) {
if(a == 5) {
if(a == 6) {
if(a == 7) {
x = 1;
}
else {
x = 0;
}
}
}
}
}
}
return 0;
}
	Compilación exitosa	Compilación exitosa

#26
int def main() {
int x = 5;
int a = 0, b = 3;
while(1 == 1) {
 	
}
return 0;
}
	Compilación exitosa	Compilación exitosa

#27
int def main() {
int x = 5;
int a = 0, b = 3;
while(1 == 1) {
int f = 5;
if(f == 5) {
unaFuncion(f);
}
else{
while(f < 10) {
f += 1;
}
}
	Compilación exitosa	Compilación exitosa

} return 0; }		

#28
int def main() {
int x = 5;
int a = 0, b = 3;
iter(0 to 10) {
imprimirVariable(a);
a += 1;
}
return 0;
}
	Compilación exitosa	Compilación exitosa

#29
int def main() {
int x = 5;
int a = 0, b = 3;
int cont = 0;
iter(a to 100000) {
imprimirVariable(a);
a += 1;
int aux;
aux %= 2;
if(aux == 0) {
cont += 1;
}
}
return 0;
}
	Compilación exitosa	Compilación exitosa

#30
const int pi = 456;
const chr c = "a";
int def main() {
return 0;
}
	Compilación exitosa	Compilación exitosa

#31
const int pi = 456;
const chr c = "a";
const dec ddd = 23.1;
const bool xr = true;
const str cadena = "holaaaaa";
int def main() {
return 0;
	Compilación exitosa	Compilación exitosa

} 		

#32
const int pi = 456;
int x = 5;
int def main() {
  return 0;
}
	Compilación exitosa	Compilación exitosa

#33
// Declaración de constantes
const int pi = 456;
// Declaración de variables
int x = 5;
dec y = 5.5;
bool z = false;
chr a = "c";
str b = "hola mundo";
int def main() {
  return 0;
}
	Compilación exitosa	Compilación exitosa

#34
// Funciones
int def miPrimerFuncion() {
  x = 1;
  return x;
}
int def main() {
  return 0;
}
	Compilación exitosa	Compilación exitosa

#35
// Funciones
int def miPrimerFuncion() {
  x = 1;
  return x;
}
int def otraFuncion(int a) {
  x += 2 + a;
  return x;
}
int def main() {
  return 0;
}
	Compilación exitosa	Compilación exitosa

#36	Compilación	Compilación

// Constantes
const int ab = 2;
const chr s = "d";
// Variables
int xar = 34555;
// Funciones
int def miPrimerFuncion() {
  x = 1;
  return x;
}
int def otraFuncion(int a) {
  x += 2 + a;
  return x;
}
// las funciones siempre deben retornar algo y tener al menos una proposición
bool def esFuncion() {
  ab += ab;
  if(ab == 44) {
  ab = 45;
  }
  return true;
}
int def main() {
  while(5 < 10) {
  if(x == 1) {
  x = otraFuncion();
  }
  }
  return 0;
}
	exitosa	exitosa

#37
const int a = 45;
int variable = 353535;
chr def unaFuncion(int a) {
  variable += 34;
  a = variable + a;
  return a;
}
int def main() {
  return 0;
}
	Compilación exitosa	Compilación exitosa

Historia - correcciones.

#4 Después de comparar el flujo esperado vs el flujo que sucedía, se determinó que el contador se adelantaba uno de más en aux18(), por lo tanto, dentro del mismo se disminuye.

#11 Se cambió la parte de proposición, definiendo 3 tipos, llamada a función simple, asignación simple a partir del valor de retorno de una función y asignación a una variable con expresión, la asignación en este último caso permite (+=, ,-=, *= ,...).

#13 Se perdía el flow por el contador, para solucionarlo fue necesario una disminución en aux33().

#14 Se crea una función postGetToken() ya que los tokens “**=” y “_/=” no se extraen como tal, sino que el signo igual (=) es separado, entonces se modificó aux32() y proposición para considerar ese detalle.

#18 Igual que en anteriores errores se ha solucionado con un decremento en aux36().

#20 En aux30() y aux29() se estaba aumentando inneseriamente el contador.

#28 En proposición para iter, faltaba obtener el token una posición adelante.

#31 Se estaba reconociendo a true y false como un identificador cuando se esperaba un boolean, entonces se cambia el orden en que se comprueba el tipo qué es, priorizando boolean para evitar el problema de tomarlo como identificador.

#32 En aux9() faltaba la recursión para seguir buscando más declaración de funciones, variables o el comienzo de la función de punto de entrada main().

#34 en aux14() no se estaba leyendo el punto y coma al final.

#36 En aux13() hacía falta un decremento.

#37 En la parte *a = variable + a;* que corresponde a una proposición de la forma de asignación de valor a variable se estaba confundiendo con la llamada a una función.