**CANTIDAD DE HOJAS:**

|  | | | **UNIVERSIDAD DE BELGRANO** | | |
| --- | --- | --- | --- | --- | --- |
| **Facultad de Ingeniería y**  **Tecnología Informática**  LENGUAJES DE PROGRAMACIÓN (0050100023LENGP)  Profesor: Mag. Ing. Pablo Pandolfo | | |
| Examen Recuperatorio octubre 2020  ALUMNO: LU: FECHA:  CARRERA: | | | | | |
| **NOTA: EL EXAMEN ESCRITO ES UN DOCUMENTO DE GRAN IMPORTANCIA PARA LA EVALUACIÓN DE LOS CONOCIMIENTOS ADQUIRIDOS, POR LO TANTO, SE SOLICITA LEER ATENTAMENTE LO SIGUIENTE:**   * Responda claramente cada punto, detallando con la mayor precisión posible lo solicitado. * Sea prolijo y ordenado en el desarrollo de los temas. * Sea cuidadoso con las faltas de ortografía y sus oraciones. * No desarrollar el examen en lápiz. * Aprobación del examen: Con nota mayor o igual a 4 (cuatro) * Condiciones de aprobación: 60% correcto. * Duración de examen: 3 horas. | | | | |  |
|  |  | Ejercicio 1 [2 puntos]: Resuélvase mediante programación lógica el siguiente problema: Los puntos del plano se representan mediante punto(X, Y), donde X e Y son níumeros, y las líneas del plano mediante linea(P1, P2), donde P1 y P2 son los puntos extremos de la misma. Definir las reglas vert(L) y horiz(L) que se verifiquen si la línea L es veritical u horizontal. Por ejemplo:  ?- vert(linea(punto(1, 2), punto(1,3))).  Yes  ?- vert(linea(punto(1, 2), punto(4, 2))).  No  **vert(linea(punto(X, Y), punto(X, Y1))).**  **horiz(linea(punto(X, Y), punto(X1, Y))).**  􏰻􏱓􏱀􏰂􏰞􏰆  Ejercicio 2 [2 puntos]: Diséñese una GIC en notación BNF para describir la sintaxis de la sentencia que crea una base de datos:  create database NOMBRE\_BASE  create table EJM\_1(campo\_1 as char, campo\_1\_2 as numeric, campo\_aux as boolean, …);  create table EJM\_2…  …  end create   * Los nombres de base, tablas y campos deben comenzar con una letra y pueden continuar con letras, números o guiones bajos ( \_ ). No puede haber más de dos guiones bajos seguidos. No puede terminar con guión bajo. * Se puede crear la base de datos vacía (sin contener tablas). * No se puede crear tablas sin al menos un campo. * El campo debe tener definido su tipo, luego del símbolo terminal as. * Son símbolos terminales: create, database, end, table, as, char, numeric, boolean, (, ), ;,   **<BD> ::= create database <ID> <TBs> ; end create | create database <ID> end create**  **<TBs> ::= <TB> | <TBs> ; <TB>**  **<TB> ::= create table <ID> ( <FD> )**  **<FDs> ::= <FD> | <FDs> , <FD>**  **<FD> ::= <ID> as <TP>**  **<TP> ::= char | numeric | boolean**  **<ID> ::= <L> | <ID> <L> | <ID> <D> | <ID> <G> <L> | <ID> <G> <D>**  **<L> ::= a |…| z**  **<D> ::= 0 |...| 9**  **<G> ::= \_**  Ejercicio 3 [2 puntos]: Diséñese el Diagrama de Sintaxis para la siguiente producción:  <Termino> ::= <Factor> {(\* | div) <Factor>}\*  Diagrama  Descripción generada automáticamente  Ejercicio 4 [2 puntos]: Demuéstrese mediante lambda calculus que la expresión **(+ 2 3)** sabiendo que + (máquina suma) es λa. λb. λc. λd. acbcd y 2 es λe. λf. e(ef) y 3 es λg. λh. g(g(gh))  **((λa. λb. λc. λd. acbcd 2) 3)**  **(λb. λc. λd. (2c) bcd) 3)**  **λc. λd. (2c) 3 c d**  **λc. λd. (λe. λf. e(ef) c) 3 c d**  **λc. λd. (λf. c(cf)) 3 c d**  **λc. λd. (c(c 3)) c d**  **λc. λd. c(c (λg. λh. g(g(gh)))) c d**  **λc. λd. c(c (λh. c(c(ch)))) d**  **λc. λd. c(c (c(c(cd))))**  Ejercicio 5 [2 puntos]: Grafíquese el Árbol de Análisis Sintáctico para la palabra:  let val si = giroizq(b) in apila (si, giro(b)) end  GIC:  <expresion> ::= a | b | giro ( <expresion>) | costura (<expresion>, <expresion>) | let <declaraciones> in <expresion> end | <id> ( <argreal> ) | <id>  <declaraciones> ::= <declaracion> | <declaracion> <declaraciones>  <declaracion> ::= fun <id> ( <arg-formales> ) = <expresion> | val <id> = <expresion>  <arg-formales> ::= <id> | <id>, <arg-formales>  <arg-reales> ::= <expresion> | <expresion>, <arg-reales> | |  |