1. Dada una pila y un valor I, desarrollar un procedimiento que elimine los 2 primeros nodos de la pila y deje el valor I como primero. (Definir parámetros y codificar).

2. Dada una pila y un valor I, desarrollar un procedimiento que inserte I como tercer valor de la pila. (Definir parámetros y codificar).

3. Idem ejercicio 60 pero retornando un parámetro con valor 'S' o 'N' según haya sido exitoso o no el requerimiento. (Definir parámetros y codificar).

4. Dada una pila y dos valores X e I, desarrollar un procedimiento que inserte el valor X en la posición I de la pila si es posible. (Definir parámetros y codificar).

5. Dada una pila y un valor X, desarrollar un procedimiento que inserte el valor X en la última posición de la pila y la retorne. (Definir parámetros y codificar).

6. Dada una pila y dos valores X e Y, desarrollar un procedimiento que reemplace cada valor igual a X que se encuentre en la pila por el valor Y retornando la pila modificada. En caso de no haber ningún valor igual a X retornar la pila sin cambio. (Definir parámetros y codificar).

7. Definir una función INVERSA que evalúe dos conjuntos de caracteres separados por un punto y retorne True si los conjuntos son inversos (ej: ABcDe.eDcBA) o False si no lo son. Los conjuntos deben ingresarse por teclado. (Definir parámetros y codificar).

8. Desarrollar un procedimiento que ingrese por teclado un conjunto de Apellidos y Nombre de alumnos y los imprima en orden inverso al de ingreso. (Definir parámetros y codificar).

9. Dada una pila desarrollar un procedimiento que ordene la misma de acuerdo al valor de sus nodos y la retorne. Solo se deben usar pilas. (Definir parámetros y codificar).

10. Dada una cola (nodo = registro + puntero), desarrollar y codificar un procedimiento que elimine 2 nodos de la misma (indicar con un parámetro 'S'/'N' si ello fue, o no posible)

11. Dada una cola (nodo = registro + puntero), desarrollar y codificar una función que devuelva la cantidad de nodos que tiene.

12. Dadas dos colas COLA y COLB (nodo = registro + puntero), desarrollar y codificar un procedimiento que genere una única cola COLAB a partir de ellas. (Primero los nodos de COLA y luego los de COLB).

13. Dada una cola (nodo = registro + puntero), imprimirla en orden natural si tiene más de 100 nodos, caso contrario imprimirla en orden inverso.

14. Dadas dos colas COLA y COLB (nodo = registro + puntero), desarrollar y codificar un procedimiento que genere otra cola COLAB por apareo del campo ARRIBO del

registro (define orden creciente en ambas). Nota: COLA y COLB dejan de ser útiles después del apareo.

15. Dado un archivo de registros de alumnos, donde cada registro contiene: a) Apellido y Nombre del alumno (35 caracteres) b) Número de legajo (7 dígitos) c) División asignada (1 a 100) ordenado por número de legajo, desarrollar el algoritmo y codificación del programa que imprima el listado de alumnos por división, ordenado por división y número de legajo crecientes, a razón de 55 alumnos por hoja.

16. Idem pero el listado de alumnos por división debe realizarse ordenado creciente por división y decreciente por número de legajo.

17. Idem anterior pero considerando que las divisiones asignadas son 100 y se identifican con un código de 4 caracteres.

18. Dado un arreglo de N (< 30) colas (nodo = registro + puntero), desarrollar y codificar un procedimiento que aparee las colas del arreglo en las mismas condiciones que las definidas en el Ejercicio 14. Nota: Retornar la cola resultante y no mantener las anteriores.

19. Dada una lista (nodo = registro + puntero), desarrollar y codificar una función que devuelva la cantidad de nodos que tiene.

20. Dadas dos listas LISTA y LISTB (nodo = registro + puntero), desarrollar y codificar un procedimiento que genere una única lista LISTC a partir de ellas. (Primero los nodos de LISTA y luego los de LISTB).

21. Dada una LISTA (nodo = registro + puntero), imprimirla en orden natural si tiene más de 100 nodos, caso contrario imprimiría en orden inverso.

22. Dadas dos listas LISTA y LISTB (nodo = registro + puntero), desarrollar y codificar un procedimiento que genere otra lista LISTC por apareo del campo LEGAJO del registro (define orden creciente en ambas). Nota: LISTA y LISTB dejan de ser útiles después del apareo).

23. Dado un archivo de registros de alumnos ARCHA sin ningún orden, donde cada registro contiene: a) Apellido y Nombre del alumno (34 caracteres) b) Número de legajo (6 dígitos) c) División asignada (1 a 100) Se debe desarrollar el algoritmo y codificación del programa que genere un archivo ARCHL igual al anterior pero ordenado por número de legajo. Nota: Memoria estática 64 Kb; dinámica suficiente si ningún nodo ocupa más de 12 bytes.

24. Dado un archivo de registros de alumnos ARCHA sin ningún orden donde cada registro contiene: a) Apellido y Nombre del alumno (34 caracteres) b) Número de legajo (6 dígitos) c) División asignada (3 dígitos) Se debe desarrollar el algoritmo y codificación del programa que imprima el listado de alumnos por división ordenado por división y número de legajo crecientes, a razón de 55 alumnos por hoja.