Práctico 1

Objetivos

- Introducir los conceptos de TAD Pila, Cola y Lista.
- Ejercitar el uso de estos TADs a partir de una especificación dada.
- Resolver problemas utilizando estos TADs.

Ejercicio 1 TAD Pila de enteros

Considere el siguiente conjunto de operaciones que especifican al **TAD Pila** de enteros.

```
/* Crea la Pila vacía */
Pila crearPila();

/* Inserta el elemento x en la cima de la Pila p. x queda como el último elemento apilado. */
Pila apilar(int x, Pila p);

/* Verifica si la Pila p está vacía */
bool esVacia(Pila p);

/* Retorna el elemento de la cima de la Pila p (último elemento apilado).
Pre : !esVacia(p) */
int cima(Pila p);

/* Retorna la Pila p sin el elemento de la cima (último elemento apilado).
Pre : !esVacia(p) */
Pila desapilar(Pila p);
```

Figura 1: Especificación del TAD Pila de enteros

Utilizando las operaciones presentadas implemente las siguientes operaciones:

(a) `imprimir`: dada una Pila *p* imprime los elementos de la misma.

(b) `cantidad`: dados un entero *x* y una Pila *p*, cuenta las ocurrencias del entero *x* en la Pila *p*.

(c) `cambiar`: dados dos enteros *x* e *y*, y una Pila *p*, retorna el resultado de cambiar el elemento *x* (cada vez que aparece) por el elemento *y* en la Pila *p*.

(d) `iguales`: dadas dos Pilas *p1* y *p2*, verifica si son iguales.

Ejercicio 2 Tad Cola de naturales

Considere el siguiente conjunto de operaciones que especifican al **TAD Cola** de naturales.

```
/* Crea y retorna la cola vacia. */
Cola crearCola();
```

```
/* Inserta el elemento x al final de la cola c. */
Cola encolar(unsigned int x, Cola c);

/* Retorna TRUE si c es vacia. */
bool esVacia(Cola c);

/* Retorna el elemento que se encuentra al comienzo de la cola c. Pre : !esVacia(c)*/
unsigned int frente(Cola c);

/* Retorna la cola resultado de borrar el primer elemento de c. Pre : !esVacia(c) */
Cola desencolar(Cola c);
```

Utilizando las operaciones presentadas implemente las siguientes operaciones:

(a) `maximo`: dada una Cola *c* no vacía, retorna su máximo elemento.

(b) `estaOrdenada`: dada una Cola *c*, verifica si está ordenada de menor a mayor.

(c) `cambiar`: dados dos enteros *x* e *y*, y una Cola *c*, retorna el resultado de cambiar el elemento *x* (cada vez que aparece) por el elemento *y* en la Cola *c*.

(d) `merge`: dadas dos Colas ordenadas *c1* y *c2*, genera una nueva cola intercalando ordenadamente ambas colas.

Ejercicio 3 Tad Lista de naturales

Considere el siguiente conjunto de operaciones que especifican al **TAD Lista** de naturales.

```
/* Crea la lista vacía. */
Lista crearLista();

/* Verifica si la lista esta vacía. */
bool esVacia(Lista l);

/* Dado un natural p, retorna TRUE si y solamente si, la lista está definida en la posición p, considerando a las posiciones a partir de 1. */
bool estaDefinida(unsigned int p, Lista l);

/* Inserta un elemento x en la posición p de la lista, considerando a las posiciones a partir de 1. Si la lista tiene longitud m y p es menor o igual a m, inserta x en la posición p y desplaza en una posición los elementos que estuvieran en las posiciones siguientes. Si la lista tiene longitud m y p es mayor a m, lo inserta en la posición m+1.
Precondición: p > 0*/
Lista insertar(unsigned int x, unsigned int p, Lista l);

/* Retorna el elemento en la posición p de l.
Pre: estaDefinida (p, l) */
unsigned int obtener(unsigned int p, Lista l);

/* Dado una posición p, elimina de la lista el elemento en la posición p. Si la posición no está definida, la operación no tiene efecto. Si la posición está definida, elimina el elemento en dicha posición y desplaza en una posición los elementos que estuvieran en las posiciones siguientes (contrae la lista). */
```

```
Lista eliminar(unsigned int p, Lista l);
```

Utilizando las operaciones presentadas implemente las siguientes operaciones:

(a) `descartar` que retorna la lista resultado de no tomar los primeros i elementos. Si la lista l tiene i o menos de i elementos devuelve la lista vacía.

(b) `largo` que dada una lista *l*, retorna la cantidad de elementos de la misma.

(c) `insertarFinal` que inserta un elemento al final de la lista.

(d) `unir` que agrega la lista p al final de la lista l.

(e) `invertir` que dada una lista *l*, retorna el resultado de invertirla.

(f) `cambiar` que retorna la lista resultado de cambiar el valor x (cada vez que aparece) por el valor y en la lista l.

Ejercicio 4 Combinación de TADs

Utilizando los TADs Lista, Pila y Cola de los ejercicios anteriores, implemente las siguientes operaciones.

(a) `invertirCola` que dada una cola c, retorne la cola c invertida.

(b) `invertirPila` que dada una pila p, retorne la pila p invertida.

(c) `removerDetrasCola` que dado una cola c y un elemento x, remueve todos los elementos que se encuentran detrás de x en la cola (mas nuevos que x).

(d) `removerDebajoPila` que dado una pila p y un elemento x, remueve todos los elementos que se encuentran debajo de x en la pila (mas antiguos que x).

(e) `ordenarPila` que dada una pila p retorna una nueva pila con los elementos de p ordenados de menor a mayor, con el elemento menor en el tope de la pila.

Ejercicio 5 Ejercicio de examen (Diciembre 2018)

Se desea saber si una pila y una cola tienen exactamente los mismos elementos y fueron insertados en ellas exactamente en el mismo orden. Para esto implemente la función `mismosElementos` usando las operaciones de los TADs `Pila` y `Cola`. Al terminar, `p` y `c` deben quedar vacías. Los elementos de `Pila` y `Cola` son de un tipo genérico que admite los usuales operadores de comparación.

```
bool mismosElementos(Pila p, Cola c);
```

No se deben hacer comparaciones innecesarias. No se pueden usar funciones auxiliares.