Este es el segundo examen parcial del curso *Programación Imperativa Modular (PIMO)*, 2015-1. El examen tiene **??** preguntas; otorga un total de **??** puntos y **??** de bono. El examen es *individual* y no es permitido el uso de libros, apuntes ni equipos electrónicos. En el desarrollo del examen, el diseño de los algoritmos y las estructuras de datos debe presentarse usando el lenguaje de programación Python.

Nombre y código: _____

Run LaTeX again to produce the table

1. Considere el tipo de datos `queue` de colas que se presenta a continuación usando la sintaxis del lenguaje de programación Python:

```
class queue(object):
  def __init__(self,cap=100):
    self.__a = [ None for i in range(cap) ]
    self.__front,self.__back,self.__size = 0,0,0
```

   El contenedor de la estructura es un "arreglo" `a` cuya capacidad inicial está determinada por el parámetro opcional `cap`. Los atributos `front` y `back` representan índices del frente y al final de la cola, respectivamente. El atributo `size` indica la cantidad de elementos almacenados en la cola. Use esta especificación para:

   (a) (3 puntos) Diseñar una operación `enqueue(x)` que inserte el valor `x` al final de la cola en orden $O(1)$.

   (b) (3 puntos) Diseñar una operación `deque()` elimine y retorne el elemento en frente de la cola en orden $O(1)$.

   (c) (4 puntos) Diseñar un método `resize(n)` que cambie el tamaño del contenedor `a` a `n` en orden $O(n)$.

   Documente cada uno de los métodos e identifique casos de error y repórtelos al usuario por medio de aserciones o excepciones. Por ejemplo, el método `resize` no debería afectar la cantidad de elementos en la cola.

2. Considere el tipo de datos `dlist` de listas doblemente encadenadas que se presenta a continuación:

```
class dlist:
  def __init__(self):
    self.__sentinel = node()
    self.__size = 0
```

```
class node:
  def __init__(self,prv=None,nxt=None,val=None):
    self._prev = prv
    self._next = nxt
    self._value = val
```

   Internamente, `dlist` usa el tipo de datos `node` para representar los valores en la lista, además de un centinela de tipo `node`. Un objeto de tipo `node` referencia a otros objetos del mismo tipo (atributos `prev` y `next`) y tiene un valor (atributo `value`). Use estas estructuras de datos y el desarrollo de *listas doblemente encadenadas con centinela* visto en clase para:

   (a) (3 puntos) Diseñar una operación `insert_last(x)` para `dlist` que inserta el valor `x` como último elemento de una lista.

   (b) (3 puntos) Diseñar una operación `remove_first()` para `dlist` que elimina el primer elemento de una lista.

   (c) (4 puntos) Diseñe una operación `shift()` para `dlist` que rota una posición hacia la derecha una lista en orden $O(1)$. Por ejemplo, si la lista $1, 2, 3$ se rota, entonces la lista resulta en $3, 1, 2$. Tenga en cuenta que rotar la lista vacía resulta en la lista vacía.

   (d) (5 +) Diseñar una operación `rotate(n)` para rotar n posiciones hacia la derecha una lista en orden $O(size)$.

   Explique brevemente cada una de las operaciones diseñadas.

3. Andrés y Beatriz cuentan con una extensa colección de CDs, en donde cada disco está identificado con un número entero. Andrés y Beatriz quieren regalar algunos discos de su colección. En particular, ellos quieren regalar aquellos discos que están en ambas colecciones.

   Entrada: $a[0..m)$ y $b[0..n)$, $m \geq 0$ y $n \geq 0$, arreglos de enteros entre -10^{12} y 10^{12} representando los identificadores de los discos de Andrés y Beatriz.

   Salida: cantidad de discos repetidos entre las colecciones de Andrés y Beatriz.

   Por ejemplo, si $a = [1, -1000, 500]$ y $b = [-1000, 100, 1, 6]$ la respuesta es 2. Puede suponer que, individualmente, cada uno de los arreglos a y b no tiene elementos repetidos.

   (a) (2 puntos) Identifique una estructura de datos estudiada en clase que permita resolver eficientemente el problema dado; explique su respuesta.

   (b) (8 puntos) Diseñe un algoritmo que use la estructura de datos identificada en la parte (a) y que resuelva el problema dado en orden temporal *promedio* $O(n + m)$; justifique su respuesta.

   (c) (10 +) Diseñe la estructura de datos identificada en la parte (a).

4. El pequeño Juan se ha metido en problemas por jugar a escondidas con los anillos de su mamá: untó pegante a los anillos y luego los tiró al piso. Algunos de los anillos cayeron sobre otros anillos, quedando pegados, y el pegante no caerá fácilmente con agua. De manera sorpresiva, ninguno de los anillos ha quedado pegado al piso. Aún con ganas de jugar, Juanito ha decidido recoger primero aquel grupo con la mayor cantidad de anillos pegados entre sí. La intención es que Usted ayude a Juanito a determinar cuál es ese grupo de anillos.

   Entrada: $r[0..n)$, $x[0..n)$, $y[0..n)$, $n \geq 0$, arreglos de números reales tales que $r[i]$ y $(x[i], y[i])$ indican el radio y la coordenada sobre el piso del centro del i-ésimo anillo, con $0 \leq i < n$.

   Salida: máxima cantidad de anillos pegados.

   Puede suponer que cada anillo es "infinitamente" delgado y que se pega a otro sii se tocan en al menos un punto.

   (a) (10 puntos) Diseñe una estructura de datos que permita resolver eficientemente el problema dado; justifique su respuesta.

   (b) (2 puntos) Dados dos anillos $0 \leq i < j < n$, proponga una fórmula que permita determinar si los anillos i y j están pegados.

   (c) (8 puntos) Diseñe un algoritmo que resuelva el problema dado eficientemente. Determine la complejidad temporal de su algoritmo, justificando su respuesta.