Nombre y Apellido:

Examen Parcial 1

1. Considere la siguiente definición que implementa una Cola:

```
#define N 50
typedef struct _Cola {
        int datos[N];
        int posicionAAgregar, posicionAEliminar;
} Cola;
```

Se pide que escriba:

a) una inicialización para `posicionAAgregar` y `posicionAEliminar`.

b) tomando en cuenta los valores del punto anterior, escriba una función `enqueue` que dada una `Cola` y un dato lo agrega a la misma (si hay lugar):

```
void enqueue(Cola* p, int datoAAgregar);
```

c) tomando en cuenta los valores del punto a, escriba una función `dequeue` que dada una `Cola` remueve el elemento correspondiente y lo retorna:

```
int dequeue(Cola* p);
```

2. Considere las siguientes definiciones:

```
typedef void (*FuncionDestructora)(void *dato);
typedef void *(*FuncionCopia)(void *dato);
typedef void (*FuncionVisitante)(void *dato);
typedef int (*FuncionComparar)(void *dato1, void *dato2);
typedef int (*Predicado) (void *dato);

typedef struct _GNode {
  void *data;
  struct _GNode *next;
} GNode;

typedef struct _GNode *GList;

// Devuelve una lista vacia.
GList glist_crear();

// Destruccion de la lista.
void glist_destruir(GList list, FuncionDestructora);

// Determina si la lista es vacia.
int glist_vacia(GList list);

// Agrega un elemento al inicio de la lista.
GList glist_agregar_inicio(GList list, void *data, FuncionCopia);
```

```
// Busca un elemento en la lista.
int glist_buscar(GList list, void* data, FuncionComparar);

// Recorrido de la lista, utilizando la funcion pasada.
void glist_recorrer(GList list, FuncionVisitante);

typedef struct _Persona {
  char* nombre;
  char* telefono;
  int edad;
} Persona;
```

a) Escriba una función que, dada una lista (`GList`) y un entero positivo `i`, retorne una lista formada por aquellos nodos que estén en las posiciones `0`, `i`, `2×i`, `3×i`, etc. La lista resultante **no** debe contener copias físicas de los datos.

b) Considere que se tiene definida una lista de estructuras `Persona` llamada `listaPersonas`. Escriba una llamada a la función anterior para obtener aquellas posiciones de `listaPersonas` que son múltiplo de 3.