Ejercicio 1. Indique si el siguiente código tiene algún tipo de problema. De encontrarlo, explique cuál es. En cualquier caso justifique su respuesta.

```
Queue q1 = emptyQ();
Queue q2 = q1;
destroyQueue(q2);
destroyQueue(q1);
```

Ejercicio 2. Describa, para cada línea del siguiente programa, cuánto espacio de memoria stack y heap se reserva, y cuál es el contenido de cada variable involucrada.

```
int y;
char* a2 = new char;
bool* b7 = new bool[60];
```

Ejercicio 3. Escriba una función que reciba una stack, y cuente la cantidad de elementos iguales al mínimo elemento.. Esta función NO debe modificar la estructura recibida por parámetro.

Ejercicio 4. Escriba el propósito del siguiente programa, e indique su costo operacional suponiendo que el árbol NO está balanceado. Justifique su respuesta.

```
void f(Tree t) {
  if(isEmptyT(t)) {
    return false;
  } else {
    return rootT(t) == 37 || f(leftT(t));
  }
}
```

Ejercicio 5. En pocas palabras, indique una ventaja de usar un lenguaje como C++ por sobre Haskell.

Ejercicio 6. Complete las partes faltantes del siguiente programa, sin modificar las existentes, de forma tal de cumplir con el propósito especificado.

```
Propósito: Dado un array de bool y su tamaño, devuelve otro array sin los primeros 42 elementos del a
Precondición: El array posee al menos 42 elementos.
bool* sinLosPrimeros42Elementos(bool* array, int tamanioDelArray) {
  ...
  for(int i = 0; i < tamanioDelArray; i++) {
    ...
  }
  ...
}
```

Ejercicio 7. Completar los tipos y expresiones faltantes en la siguiente representación de un Grupo de alumnos como una cadena de punteros.

```
struct NodoA {
  string nombre;
```

```
  ... siguiente;
};

struct GrupoSt {
  ... primero;
};

typedef ...* GrupoDeAlumnos;

// Propósito: Devuelve un nuevo grupo vacío
... nuevoGrupo() {
  ... nG = new GrupoDeAlumnos;
  nG->primero = ...;
  return nG;
}

// Propósito: Agrega un elemento nuevo, al principio de la cadena
void agregarAlGrupoDeAlumnos(string nombre, GrupoDeAlumnos grupo) {
  NodoA* nodo = new ...;
  nodo->nombre = nombre;
  ... = grupo->primero;
  grupo->primero = ...;
}

// Propósito: Libera la memoria ocupada por la estructura
void destroyGrupoDeAlumnos(GrupoDeAlumnos grupo) {
  NodoA* actual = grupo->primero;
  while(actual != ...) {
    ...
    actual = ...
  }
  delete ...;
}
```

Ejercicio 8. Un leak de memoria es un sector de memoria reservado al que el programa ha perdido referencia. Escriba un programa que produzca un leak de memoria, utilizando la interfaz del tipo Stack. Justifique su respuesta.