Estructuras de Datos.
Relación de Problemas: STL

Rosa Mª Rodríguez Sánchez.

1. Definir una función que permita invertir un objeto de tipo list. Los elementos que contiene la lista son enteros

   Invertir(const list<int> & lsource, list<int>& ldestino)

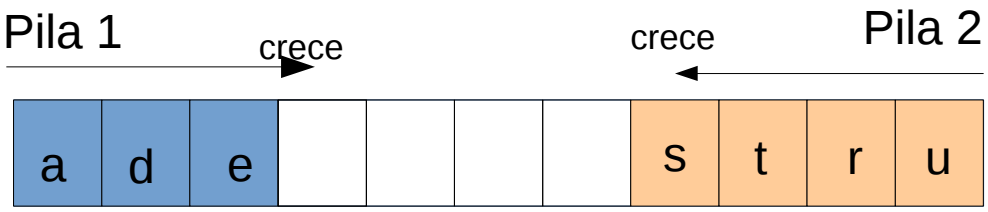
2. Suponer que tenemos información de los alumnos que desean acceder a una carrera junto con su nota de selectividad.

```
struct alumno{
   char dni[9];
   string nombre;
   string apellidos;
   string correo;
   double nota_selectividad;
};
```

   Definir una función que obtenga una cola de prioridad (priority_queue) con la información de todos los alumnos, de manera que la prioridad se define de mayor a menor valor de selectividad. Así la función sería:

   void ObtenerPrioridad (const list<alumno> & alumnos, prority_queue<alumno> & pq);

   alumnos almacena la información de todos los alumnos.
3. Dada la clase list instanciada a enteros, crear una función que elimine los elementos pares de la lista. Para implementarla hacer uso de iteradores.
4. Definir el T.D.A DoblePila que contiene dos pilas de caracteres usando un único vector para ello. Así una de las pilas empieza a poner sus datos desde la posición 0 hacia adelante y la otra desde el máximo espacio reservado hacia atrás. Un esquema de este tipo se puede ver en la siguiente figura:



   Dar una representación de este TDA e implementar los siguiente métodos:
   1. char Tope(key_pila kp): devuelve el tope de la pila con código kp. Key_pila puede ser un enumerado con dos valores Pila1 y Pila2.
   2. Void Poner(key_pila kp,char c): inserta un nuevo elemento en el tope de la pila kp. Si ya no tenéis espacio sobre el vector deberéis redimensionar el vector.
   3. Void Quitar(key_pila kp): elimina del tope de la pila kp.
   4. Bool Vacia(key_pila kp). Indica si la pila kp está vacía.

5. Suponer que tenemos el ***T.D.A Matriculas*** que almacena parejas de dni de un alumno y el código de asignatura en el que está matriculado. Escoger la representación más adecuada para este tipo de dato, de forma que si consulta por los alumnos que están matricualdos en una asignatura concreta sea lo más eficiente posibile. Y sí además, pregunto por las asignaturas que un determinado alumno está matriculado también sea eficiente. Implementar el método insertar, Borrar que añade una nueva matricula y borra una matricula existente

   void Matriculas::Insertar(const string& dni,const string &cod_asig);
   void Matriculas::Borrar(const string &dni,const string &cod_asig);

6. Siguiendo con el ***T.D.A Matriculas*** del ejercicio anterior:
   1. Definir un iterador que permita iterar sobre todas las mátriculas.
   2. Implementar el método que permite consultar todos los alumnos (dni) matriculados en una asignatura:

      list<string> Mariculas::GetAlumnos(const string &cod_asig);
   3. Implementar el método que permite consultar todos los alumnos (dni) matriculados en una asignatura:

      list<string> Mariculas::GetAsignaturas(const string &dni);