A. Contexto

Para llevar a cabo su propósito, buscadores como Google o Bing deben indexar enormes cantidades de páginas web. En el caso de Google, la cifra exacta se mantiene en secreto, pero se estima que supera los 400 mil millones. Por tanto, además de precisar una importante infraestructura física y un costoso soporte hardware, se hace imprescindible usar estructuras de datos muy eficientes para guardar la información y diseñar algoritmos muy rápidos para llevar a cabo las búsquedas sobre esa enorme cantidad de información.

El objetivo de esta práctica es crear un buscador de páginas web que haga uso de una técnica apropiada para almacenar el contenido de las páginas y que permita realizar distintos tipos de búsqueda de manera eficiente.

B. El Problema

Analizar, diseñar e implementar un buscador de páginas web por contenido. El programa admitirá una serie de comandos, que se leerán siempre desde la entrada estándar, produciendo el resultado en la salida estándar. Los comandos admisibles son:

- **Insertar nueva página**: indicando URL, título, relevancia y el texto contenido en la página.
- **Búsqueda por URL**: buscar una página dada su URL.
- **Búsqueda de palabra**: búsqueda sencilla de las páginas que tengan una palabra dada. Las palabras se definen como cualquier sucesión de una o más letras delimitadas por caracteres que no sean letras.
- **Buscar palabras con AND (opcional)**: listar todas las páginas donde aparecen ***todas*** las palabras dadas.
- **Buscar palabras con OR (opcional)**: listar todas las páginas donde aparezcan ***algunas*** de las palabras dadas.
- **Autocompletar (opcional)**: completar una palabra escrita a medio.

El programa diseñado deberá cumplir los siguientes requisitos:

- Las búsquedas deben ser independientes de mayúsculas/minúsculas y tildes.
- Se requiere que no se pierda ni una sola búsqueda.
- El programa tiene que estar bien diseñado para conseguir la máxima eficiencia de tiempo y de memoria (con especial hincapié en lo primero).

C. Formato de entrada

La entrada está compuesta de varios comandos. Todos los comandos están en una línea distinta, excepto la inserción que ocupará varias. Los comandos admisibles son: **i** (para insertar), **u** (búsqueda por URL), **b** (búsqueda de una palabra), **a** (búsqueda con AND), **o** (búsqueda con OR), **p** (autocompletar), **s** (salir). La entrada acabará siempre con un comando **s**. El formato de los comandos es el siguiente:

- **Insertar**: después de la **i** aparecerá un entero R que indica la relevancia que se le asigna a la página (suponer un número natural). La siguiente línea contiene la URL de la página (que será única para cada página), y la siguiente el título de la misma. A continuación viene el contenido de la página en sí. Cada página puede

ocupar una o varias líneas, y acaba con la palabra clave "FinDePagina", que es independiente de mayúsculas, minúsculas o tildes. Este puede ser, por ejemplo, un comando **i** válido:

```
i 5
http://www.um.es
Universidad de Murcia
La Universidad de Murcia del tercer milenio tiene como eje central de su actividad la consecución de la excelencia académica y científica.
Todo ello queda recogido en esta Web que pretende satisfacer las necesidades no solo de la comunidad universitaria sino de la Sociedad en general.
FINDEPAGINA
```

- **Búsqueda por URL**: después de la letra **u**, aparecerá una dirección URL que se quiere buscar. Por ejemplo:

```
u http://dis.um.es/index.html
u http://www.um.es
```

- **Búsqueda de palabra**: después de la letra **b**, aparecerá una palabra que se busca. Por ejemplo:

```
b condición
b PROGRAMACION
```

- **Búsqueda con AND (opcional)**: después de la letra **a**, aparecerá una lista de una o más palabras que se buscan. Por ejemplo:

```
a llevo la Condición
a la PROGRAMACION de Meyer
```

- **Búsqueda con OR (opcional)**: después de la letra **o**, aparecerá una lista de una o más palabras que se buscan. Por ejemplo:

```
o condicion
o TROYANO TROYA
```

- **Autocompletar (opcional)**: después de la letra **p**, aparecerá un prefijo de una palabra. Por ejemplo:

```
p autocom
p Universi
```

D. Formato de salida

Después de cada comando, se mostrará por pantalla información sobre el resultado del mismo. La salida tendrá el siguiente formato:

- **Insertar**: la salida serán 2 líneas, que contendrán: (1) el número ***consecutivo*** de la página respecto al resto de páginas introducidas, la URL, el título y la relevancia de la página insertada (separados por comas), y (2) el número total de palabras leídas, ya sean repetidas o no. Por ejemplo, la salida para el ejemplo de arriba será:

```
1. http://www.um.es, Universidad de Murcia, Rel. 5
46 palabras
```

- **Búsqueda por URL, palabra, AND y OR**: en primer lugar aparecerá el comando de búsqueda. Después, los resultados irán numerados de forma consecutiva, empezando en 1, y para cada uno se indicará: la URL, el título y la relevancia, separados por comas. Si hay varios resultados, la lista estará ordenada de acuerdo con la relevancia de la página (de mayor a menor), y en

caso de empate por título, alfabéticamente. Por ejemplo, el resultado para la búsqueda "o actividad" en el ejemplo de arriba sería:

```
o actividad
1. http://www.um.es, Universidad de Murcia, Rel. 5
Total: 1 resultados
```

- **Autocompletar**: igual que antes, en primer lugar aparecerá el comando leído en la entrada. Después aparecerá la lista de palabras y el número de resultados. Por ejemplo:

```
p Universi
1. universidad
2. universitaria
Total: 3 resultados
```

E. Fases de desarrollo

Para una correcta resolución de esta práctica, se proponen las siguientes fases de desarrollo, que marcan de manera aproximada el ritmo que se seguirá en las clases de prácticas. Se usará el juez on-line de la asignatura, que contendrá los ejercicios correspondientes a cada fase. Las fases de desarrollo son:

E1. Semanas 1, 2 y 3 (semanas del 7, 14 y 21 de octubre): resolver los problemas básicos referentes a separar palabras, normalizar el texto, el intérprete de comandos, y el tipo de datos página.

E2. Semanas 4 y 5 (semanas del 28 de octubre y 4 de noviembre): implementar tablas de dispersión para almacenar las páginas y para hacer las consultas por URL.

E3. Semanas 6 y 7 (semanas del 11 y 18 de noviembre): crear un diccionario de palabras con árboles (ya sea trie, AVL o B) para almacenar las palabras. Implementar la operación de buscar por palabra.

E4. Semanas 8 y 9 (hasta el 5 de diciembre): juntando las implementaciones anteriores, escribir el programa capaz de procesar los comandos obligatorios, conteniendo la tabla de dispersión y el árbol. Ese mismo día será la entrega final de la memoria de la práctica.

F. Documentación

Al terminar la práctica se entregará la documentación final. Se debe **documentar el resultado final de la práctica**. La memoria se entregará en formato PDF en el Aula Virtual y contendrá obligatoriamente los siguientes apartados:

1. **Portada**. Nombre de los alumnos, grupo, subgrupo y cuenta usada en Mooshak.
2. **Análisis y diseño del problema**. De forma orientativa, la documentación del análisis y diseño de la práctica debe responder de forma justificada a las siguientes preguntas:
   - ¿Qué clases se han definido y qué relación existe entre ellas? Incluir una representación gráfica de las clases.
   - ¿Qué módulos existen, qué contienen y cuál es la relación de uso entre ellos? Poner una representación gráfica de los ficheros y sus dependencias (includes).
   - ¿Cómo se hace la normalización del texto?
   - ¿Cómo es el Makefile? ¿Contiene todas las dependencias existentes?

- ¿Qué tipo de tablas de dispersión se ha usado y por qué? Justificar la decisión.
- ¿Qué función de dispersión se ha usado? ¿Se han probado varias? Hacer una comparativa de las distintas funciones que se han probado.
- Si se hace reestructuración de las tablas, explicar cómo y justificar la decisión.
- ¿Cómo se libera la tabla de dispersión?
- ¿Qué tipo de árboles se han implementado y por qué?
- ¿Cómo es la definición de la clase árbol y del nodo?
- ¿Cómo se almacenan las páginas asociadas a cada palabra en el árbol?
- Si se han implementado árboles AVL, ¿cómo se hace el balanceo?
- ¿Cómo se liberan los árboles?
- ¿Se usan variables globales en el programa final?
- Si se han usado herramientas como ChatGPT, describir de forma detallada en qué partes y cómo se han usado.

3. **Listado del código**. Incluyendo el fichero `Makefile` necesario para compilar. Listar solo el programa final (el que contiene todos los comandos).
4. **Informe de desarrollo**. Describir cómo ha sido la coordinación y el reparto del trabajo entre los miembros del grupo. Completar la tabla 1.6 de la página 27 del texto guía, que debe rellenarse mientras se hace la práctica (ver esta tabla en el Aula Virtual, en Recursos / Fragmentos del texto guía / cap1y2.pdf).
5. **Conclusiones y valoraciones personales**.

G. Evaluación de la práctica

G.1. Obligatorio

Para aprobar la práctica se requiere que:

- El programa se pueda ***compilar sin errores*** y, para todos los ejercicios, el código debe haber sido aceptado por el juez on-line de la asignatura (Mooshak). El programa estará escrito en C++, y se deberá compilar en Linux.
- El programa debe ***funcionar correctamente***, sin colgarse y produciendo ***resultados correctos*** para el conjunto de pruebas que se determinen. Para ello, el profesor puede usar (pero no está limitado a) los casos de prueba incluidos en el juez on-line de la asignatura.
- La ***memoria de la práctica*** debe contener todos los puntos indicados en el apartado F, y debe ser entregada en el plazo que se establezca. ¡La documentación entregada no debe contener *faltas de ortografía* (incluida la omisión de tildes)!

G.2. Criterios de valoración

La práctica se puntuará de acuerdo con los siguientes criterios de calidad del software:

- **Análisis y diseño**. Se valorará la calidad y adecuación del diseño y el análisis realizados, y la elección de las estructuras de datos más adecuadas para cada necesidad.
- **Modularidad.** La funcionalidad debe estar bien repartida entre los módulos. Se debe respetar la ocultación de la implementación, usar *Makefile* y compilación separada.

- **Uso del lenguaje**. El código deber ser claro, legible, robusto y eficiente. No crear procedimientos muy largos y complejos. Usar correctamente las características de C++.
- **Eficiencia**. Un aspecto fundamental en la valoración de la práctica será la eficiencia conseguida por el programa, tanto de tiempo como de uso de memoria, usando los conceptos estudiados en la asignatura.

G.3. Otras cuestiones

La práctica se deberá realizar preferiblemente en **grupos de dos alumnos**. De forma extraordinaria se permiten **grupos de un alumno**, pero no se prevé una reducción del trabajo para los mismos.

Para realizar pruebas y para la verificación de las fases de desarrollo, los profesores dejarán en la página web del juez on-line (https://mooshak.inf.um.es), dentro del concurso "AED1, 24/25. Práctica", los problemas mencionados en el apartado **E**. Cada alumno dispondrá de un usuario y clave para acceder a este sistema; el grupo deberá elegir y utilizar solo una de las cuentas para hacer los envíos al juez. La cuenta de cada alumno se puede consultar en el Aula Virtual, dentro del sitio de la asignatura, en Calificaciones, en los comentarios de la calificación Nota de Práctica.

La fecha de entrega definitiva de esta práctica es el **jueves 5 de diciembre de 2024**. El concurso del juez se cerrará ese día a las 21:00, siendo la entrega de la documentación en PDF a lo largo del mismo día, a través de la Tarea correspondiente del Aula Virtual.

AVISO IMPORTANTE

Las prácticas de todos los grupos, en todas las convocatorias y titulaciones, serán sometidas a un sistema de **detección de plagios**. Copiar la práctica de otro grupo (en todo o en parte) supondrá el suspenso fulminante de la asignatura en la convocatoria correspondiente, para todos los grupos implicados. Esto incluye el uso de sistemas generativos como ChatGPT sin que se reconozca ni documente su uso.