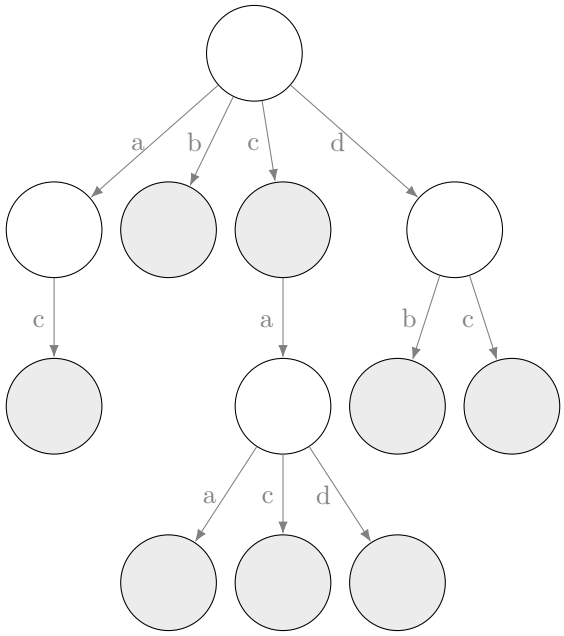
1151060 - Temas Selectos de Lenguajes de Programación
Tarea 5 - 2022-O.

Un árbol de prefijos o trie permite almacenar un conjunto de cadenas. Sea Σ el conjunto de los caracteres que pueden aparecer en las cadenas (también denominado conjunto *alfabeto*), cada nodo del trie es un nodo $|\Sigma|$-ario donde cada transición a un nodo hijo corresponde con un caracter del alfabeto. Las transiciones del nodo raíz particionan las cadenas del trie según el primer caracter, mientras que los niveles inferiores del árbol se encargan de los caracteres sucesivos. No es necesario guardar copias explícitas de las cadenas almacenadas, porque las transiciones del árbol las denotan implícitamente. El fin de una cadena debe tener una representación especial.



Ejemplo de trie para las cadenas `ac`, `b`, `c`, `caa`, `cac`, `cad`, `db`, `dc`. Los nodos sombreados son terminales de cadena. Todas las hojas son terminales, pero también puede haber terminales internos.

El código disponible en https://racc.mx/uam/home/2022-o/tslp/tarea5.cpp implementa un trie que pretende ser usado para almacenar cadenas ASCII y que provee las siguientes funciones miembro:

- `void inserta(const std::string& s);`
  Inserta la cadena `s` en el trie.
- `bool con_terminal( ) const;`
  Regresa verdadero sí y sólo si el nodo actual es un nodo que denota un terminal de cadena.
- `const trie* hijo(char c) const;`
  Si el nodo actual tiene un nodo hijo bajo la transición del caracter `c`, regresa un apuntador a dicho nodo hijo. En caso contrario, regresa nulo.

Implementa la plantilla de función `void visita(const trie& t, auto&& retrollamada);` cuya semántica debe ser la siguiente. Si el trie tiene n cadenas, entonces la función `visita` debe invocar a la retrollamada n veces como si fuera una función, pasándole en cada llamada un valor de tipo `std::string` que contenga cada una de las cadenas del trie, en orden lexicográfico ASCII.

Tu código no declarar `main` y no debe usar variables estáticas o globales. Se permite declarar tipos y funciones auxiliares, así como constantes globales en tiempo de compilación. Cada función puede declarar variables auxiliares que no superen los 500 kilobytes en total por función. Un programa que visite un trie con menos de un millón de nodos debe debe terminar su ejecución en menos de un segundo. Sólo se pueden usar los archivos de biblioteca ya presentes en el código original. Puedes consultar una página de prueba en https://racc.mx/uam/home/2022-o/tslp/tarea5.html. Deberás enviar el código fuente de tu programa desde tu cuenta institucional al formulario https://forms.gle/WneQfaPq9ob6Qocv9. Tu código será evaluado con varios casos de prueba y se espera que cumpla la semántica descrita.

Ejemplo de uso	Ejemplo de salida

```
int main( ) {
   trie t;
   t.inserta("gatito");
   t.inserta("perrito");
   t.inserta("gat");
   visita(t, [](const std::string& s) {
      std::cout << s << "\n";
   });
}
```

```
gat
gatito
perrito
```