UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructuras de Datos I

Primer cuatrimestre de 2021

# Taller de Strings

Resolver los siguientes ejercicios en la computadora. Para los mismos utilizaremos la clase string [1]. Solo está permitido operar sobre ellos utilizando `size()`, `push_back()` y `clear()`, salvo se indique lo contrario.

Los alumnos deben:

- Abrir Clion;
- Hacer click en Open;
- Seleccionar la carpeta template_alumnos;

## 1. split

```
vector<string> split(string s, char delim);
```

Separar el string según el delimitador pasado por parámetro.

Por ejemplo, `split("la casa está en orden", ' ')` debe devolver `<"la", "casa", "está", "en", "orden">`.

## 2. darVueltaPalabra

```
string darVueltaPalabra(string s);
```

Dado un string solo conformado por letras y espacios, devolver otro que resulte de dar vuelta el orden de las palabras.

Por ejemplo, `darVueltaPalabra("hola mundo")` debe devolver `"mundo hola"`.

## 3. subsecuencia

```
bool subsecuencia(string s, string t);
```

Decidir si **s** es subsecuencia de **t**. Un string **s** es subsecuencia de **t** si existen índices $i_1 < i_2 < i_3 ... < i_{|s|}$ tales que $t[i_1] = s[0],\ t[i_2] = s[1] \ldots, t[i_{|s|}] = s[|s| - 1]$

Por ejemplo, `subsecuencia("ace", "abcdec")` debe devolver **true** pero `subsecuencia("abdc", "abcecde")` debe devolver **false**.

## 4. agruparAnagramas

```
vector<vector<string>> agruparAnagramas(vector<string> v);
```

Dado un vector de strings devolver un `vector<vector<string>>` tal que se cumpla que:

- Todos los strings del vector de entrada están en exactamente un subvector resultado (y el resultado no tiene strings que no estén en la entrada);
- Cada elemento (vector) del resultado contiene solo anagramas;
- Se mantiene el orden relativo de cada palabra dentro de cada vector y el orden de cada vector responde a la primera aparición de un nuevo anagrama.

Por ejemplo, `agruparAnagramas({"ab", "cd", "ef", "ba", "ab", "dc"})` debe devolver
`{{"ab", "ba", "ab"}, {"cd", "dc"}, {"ef"}}`

[1] http://www.cplusplus.com/reference/string/string/

## 5. esPalindromo

```
bool esPalindromo(string s);
```

Decidir si un string es palíndromo, es decir, si se escribe igual en ambas direcciones.

## 6. tieneRepetidos

```
bool tieneRepetidos(string s)
```

Decidir si un string tiene algún caracter repetido.

## 7. rotar

```
string rotar(string s, int j);
```

Devolver un string que esté rotado $j$ veces hacia la derecha. Por ejemplo `rotar("hola", 7)` debe devolver `"olah"`.

## 8. darVueltaK

```
string darVueltaK(string s, int k);
```

Dado un string `s`, dar vuelta cada `k` caracteres el substring de longitud `k`. Si la cantidad de caracteres de la entrada no es múltiplo de `k`, el último substring (de longitud menor a `k`) también debe darse vuelta.

Por ejemplo, `darVueltaK("abracadabra", 3)` debe devolver `"rbaacabadar"`.

## 9. Análisis de la risa

Científicos rusos están investigando propiedades de la risa. Para ello, analizan el habla humana e identifican qué sonidos pertenecen a una risa.

Estos científicos ya hicieron un software que traduce el habla humana en texto. Consideran que la risa es una secuencia alternada de las letras "h" y "a". Por ejemplo, "ahahaha", "hah" y "a" son risas, pero "abacaba" y "hh" no.

Dado un string $s$ encontrar la longitud de la risa más larga, es decir, un substring de $s$ que sea risa y que sea más larga que el resto de las risas.

Deben implementar la función en el archivo `risas.cpp`:

```
int risaMasLarga(string s)
```

### Ejemplos

- `risaMasLarga("ahaha")` debe ser igual a 5.
- `risaMasLarga("ahahrunawayahahsofasthah")` debe ser igual a 4.
- `risaMasLarga("ahahaahaha")` debe ser igual a 5.

### Datos y Tests

En la carpeta `risas` se encuentran dos archivos de prueba como ejemplo del formato de entrada que sus funciones de E/S deben saber levantar. En el archivo `risas.cpp` hay también dos funciones que es preciso implementar para poder leer los datos. Este formato consiste en una sola línea donde cada caracter está separado por un espacio. Por ejemplo, el contenido del archivo `01.in` es:

```
a h a h a
```

La respuesta esperada para esa entrada se encuentra en el archivo `01.out` cuyo contenido es un entero indicando la risa más larga.

En la carpeta test, el archivo `risasTEST.cpp` contiene un gran número de tests que debe cumplir el código.

# 10. Problemas

## 10.1. Ayudando al abuelo Laino (Dificultad Baja)

No sabemos concretamente por qué, pero al abuelo Laino no le agrada una de las vocales de la lengua castellana. Tal vez sea para él engorroso declamarla, con certeza el tartamudeo estaba mal contemplado en épocas pasadas. En todo caso, afortunadamente esa vocal no esta presente en la palabra *abuelo*, de manera que la prole de sus vastagos no se encuentra en apuros cuando le llama afectuosamente de ese modo.

El abuelo Laino trabajó con nulo descanso durante décadas, por lo que se prepara para tomar un justo receso de sus arduas tareas. En este lapso, desea emprender una aventura atravesando parajes lejanos, para lo cual esta, ahora empacando su maleta. El abuelo Laino no desea llevar en ella objetos cuyos nombres contengan la vocal que tanto lo consterna, no vaya a ser que al verlos se vea forzado a pensar en la tan censurable letra durante su reposo. Su tarea es ayudarlo en esta labor, para lo cual deben aconsejarlo sobre cuales de los objetos que posee puede empacar.

Para ello, se debe realizar una función que dado un string s que representa el objeto a empacar, devuelva un caracter **S** en caso de que el abuelo Laino lo pueda empacar o en caso contrario un caracter **N**.

Ejemplos:

- abueloLaino(remera) = S
- abueloLaino(camisa) = N
- abueloLaino(buey) = S
- abueloLaino(abuelo) = S
- abueloLaino(revista) = N
- abueloLaino(iman) = N
- abueloLaino(calendario) = N

## 10.2. Felicitaciones Francisco (Dificultad Media)

Francisco ha finalizado una fructífera etapa de su formación en la facultad, obteniendo su doctorado en física. Esto es fundamentalmente fruto de la firmeza a la hora de formalizar los fabulosos resultados de su fantástica investigación.

Francisco ha fomentado siempre la felicidad entre sus amigos y familiares, por lo que hemos decidido firmarle una felicitación a nuestro doctor favorito. Habiendo finalizado la carta de felicitación, solo resta incorporar la fervorosa firma de los **F** firmantes. Para esto, hemos comprado dos fibrones, uno de color azul Francia, y el otro de color fucsia, con los que cada firmante escribira su nombre.

Es nuestra intención hacer foco en el titulo de doctor que acaba de obtener Francisco, y por lo tanto queremos que en nuestras firmas se vea camuflada muchas veces la abreviación de doctor ("DR"). Para esto, firmaremos utilizando ambos colores, y escribiremos algunas letras en azul y otras en fucsia, de modo tal que si Francisco lee solo las letras de color fucsia pueda leer la sigla "DR".

Nuestro objetivo es que, al leer únicamente las letras escritas en fucsia, Fidel solo lea las letras D y R de manera alternada. Por lo tanto, la primera letra de color fucsia debe ser una D, y para cada letra D que esta escrita en fucsia, la proxima letra de ese color deberá ser una R. Análogamente, para cada letra R de color fucsia la siguiente letra de ese color deberá ser una D, siendo entonces una R la ultima letra de color fucsia.

Queremos escribir las letras en fucsia de modo tal que Francisco lea las letras D y R en ese color la mayor cantidad de veces posible. Para cumplir nuestro objetivo, podemos elegir en que orden escribir nuestros nombres, y que letras escribir de cada color. Como hay muchas formas de hacer esto, les pedimos ayuda para que nos digan cual es la mayor cantidad de veces que podemos escribir las siglas "DR" de color fucsia si respetamos las reglas dadas en el párrafo anterior.

Para lograr esto, se deberá realizar una función que tome como entrada un vector de strings (representando cada uno de los amigos de Francisco) y deberá devolver un entero que representa la máxima cantidad de veces que puede aparecer la sigla DR en fucsia al firmar nuestra carta de felicitación, si escribimos las letras D y R alternadamente como se describe en el enunciado.

Ejemplos:

| Entrada de Ejemplo | Salida de Ejemplo |
|---|---|
| RAMIRO<br>AUGUSTO<br>JOAQUIN<br>JACINTO<br>NICOLAS<br>ALEJANDRO<br>DIJKSTRA<br>KAJITA<br>MCDONALD<br>SCHRODINGER | 4 |

| Entrada de Ejemplo | Salida de Ejemplo |
|---|---|
| DDD<br>RRR<br>DRDR<br>RDRD | 5 |

| Entrada de Ejemplo | Salida de Ejemplo |
|---|---|
| MELANIE<br>DAMIAN<br>RAMIRO<br>AUGUSTO<br>JOAQUIN<br>JACINTO<br>NICOLAS<br>ALEJANDRO<br>DIJKSTRA<br>KAJITA<br>MCDONALD<br>SCHRODINGER | 5 |

| Entrada de Ejemplo | Salida de Ejemplo |
|---|---|
| ABCEFG<br>HIJKLM<br>NOPQST<br>UVWXYZ | 0 |

## 10.3. Middle-Out (Dificultad Alta)

***Para este ejercicio se permite utilizar la funcion sort provista por la std.***

Dados dos strings $\boldsymbol{s}$ y $\boldsymbol{t}$ de la misma longitud $\boldsymbol{n}$, queremos saber si es posible "transformar" $s$ en $t$. Pero para esto, se deben seguir las siguientes reglas:

Pensemos que sus caracteres están numerados de **1** a **n** de izquierda a derecha (es decir, de principio a fin).

En cada movimiento se debe hacer lo siguiente:

- elegir un indice valido $i$ ($1 \leq i \leq n$).
- mover el $i$-ésimo caracter de $s$ desde su posición al principio del string, o mover el $i$-esimo caracter de $s$ desde su posición al final del string

Notemos que los movimientos no cambian el largo del string s.

Por ejemplo, si $s =$ "$test$" en un movimiento se puede obtener lo siguiente:

- Si $i = 1$ y movemos al principio, el resultado es: $test$ (el string no cambia),
- Si $i = 2$ y movemos al principio, el resultado es: $etst$,
- Si $i = 3$ y movemos al principio, el resultado es: $stet$,
- Si $i = 4$ y movemos al principio, el resultado es: $ttes$,
- Si $i = 1$ y movemos al final, el resultado es: $estt$,
- Si $i = 2$ y movemos al final, el resultado es: $tste$,
- Si $i = 3$ y movemos al final, el resultado es: $tets$
- Si $i = 4$ y movemos al final, el resultado es: $test$ (el string no cambia)

Como se dijo anteriormente, se desea que el string s sea igual al string t. Para ello, se deberá realizar una función que dados los string s y t, devuelva cual es el mínimo numero de movimientos necesario para lograrlo.

En caso de ser posible, la función debe devolver un entero que indica la mínima cantidad de movimientos necesarios. En caso de ser imposible transformar s a t, se debe devolver -1.

Ejemplos:

- middleOut(estt, test) = 1
- middleOut(tste, test) = 2

- middleOut(iredppipe, piedpiper) = 2
- middleOut(adhas, dasha) = 2
- middleOut(aashd, dasha) = 2
- middleOut(aahsd, dasha) = 3
- middleOut(a, z) = -1