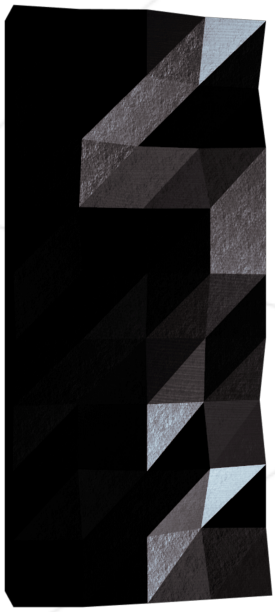

C Piscine

Rush 02

Resumen: Este documento corresponde al enunciado del rush 02 de la C piscine de 42.

Versión: 10.1

Índice general

Capítulo I

Instrucciones

- El grupo se registrará automáticamente para la evaluación. No canceles la evaluación, no tendrás una segunda.
- Toda petición de precisiones sobre los enunciados complicará los ejercicios.
- Debéis respetar el procedimiento de entrega para todos vuestros ejercicios.
- Este enunciado puede cambiar hasta una hora antes de la entrega.
- La Moulinette compila con las flags -Wall -Wextra -Werror y utiliza `cc`.
- Si vuestro programa no compila, la nota final será `0`.
- Vuestro programa debe estar escrito de acuerdo a la Norma. Si tienes archivos y/o funciones de bonus, están incluidos en esta regla y tendréis un `0` si contienen algún error de norma.
- Debéis entregar un Makefile, que compile vuestro proyecto con reglas `$NAME`, `clean` y `fclean`
- Tienes que realizar este proyecto con resto de miembros del equipo asignado y debéis presentaros todos a la evaluación a la hora acordada.
- El proyecto debe estar terminado cuando se presente a la evaluación. El propósito de la evalución es que presentéis y expliquéis vuestro trabajo en detalle.
- Cada miembro del grupo tendrá que estar perfectamente al corriente del trabajo realizado. Si elegís dividiros el trabajo, aseguraos de que todos entendéis lo que ha hecho el resto. Durante la evaluación se harán preguntas y la nota del grupo se basará en la peor explicación.
- Es tu responsabilidad reunir al equipo de trabajo. Tienes todos los medios disponibles para contactar con el resto de miembros del equipo: teléfono, e-mail, paloma mensajera, sesión de espiritismo, etc. No se aceptará ninguna excusa en lo que respecta a los problemas de grupo. La vida es injusta, pero es lo que hay.
- De todas manera, si después de haberlo intentado realmente todo no puedes contactar con un miebro de tu grupo: haced el rush y entregadlo. Intentaremos encontrar una solucion durante la evaluación. Incluso si el integrante que falta es el team leader, todos tenéis acceso al repositorio.

- Obviamente, vuestro trabajo deberá respetar La Norma. ¡Sed muy rigurosos!
- ¡Disfrutad!

Capítulo II

Introducción

Aquí tienes una receta de Tarta de nueces a la antigua:

```
Ingredientes
- Masa de hojaldre
- 3/4 barra de mantequilla sin sal
- 1 1/4 tazas de azúcar moreno ligero
- 3/4 taza de jarabe de maíz ligero
- 2 cucharaditas de extracto de vainilla
- 1/2 cucharadita de cáscara de naranja rallada
- 1/4 de cucharadita de sal
- 3 huevos grandes
- 2 tazas de nueces partidas (225 gramos)
Acompañamiento: nata montada o helado de vainilla

Preparación:
Precalentar el horno a 180º con una bandeja para hornear en la rejilla del
medio.
Sobre una superficie ligeramente enharinada, extiende la masa con un
rodillo ligeramente enharinado en una forma redonda de 30 centímetros
de diámetro y ponla en el fondo de un molde redondo para tarta de 22
centímetros de diámetro.
Recorta el borde, dejando un saliente de 1 centímetro.
Dobla el saliente por debajo y presiona ligeramente contra el borde del
molde.
Pincha ligeramente el fondo con un tenedor.
Deja enfriar hasta que esté firme, al menos 30 minutos.
Mientras tanto, derrite la mantequilla en una cacerola pequeña a fuego
medio.
Añadir el azúcar moreno, batiendo hasta que esté suave.
Retirar del fuego e incorporar el jarabe de maíz, la vainilla, la
ralladura y la sal.
Bate ligeramente los huevos en un cuenco mediano y luego añade a la
mezcla de jarabe de maíz.
Pon las nueces en el molde de la tarta y vierte la mezcla uniformemente
sobre ellas.
Hornea en una bandeja caliente hasta que el relleno esté cuajado,
```

```
entre 50 minutos y 1 hora.
Deja enfriar completamente.

Notas del cocinero:
La tarta se puede hornear con 1 día de antelación y refrigerarla.
Déjala a temperatura ambiente antes de
servir.
```

Capítulo III

Enunciado

	Ejercicio: 00
rush-02	
Directorio de entrega: *ex*00/	
Archivos a entregar: `Makefile y todos los archivos necesarios`	
Funciones autorizadas: `write, malloc, free, open, read, close`	

- Debéis crear un programa que reciba un número como argumento de entrada y lo convierta en su valor escrito.
- Nombre del ejecutable: `rush-02`
- Vuestro código fuente será compilado por el comando:

```
make fclean
make
```

- Vuestro programa puede recibir hasta 2 argumentos:
  - Si solo hay un argumento, es el valor que necesita convertir.
  - Si hay dos argumentos, el primero es el nuevo diccionario de referencia y el segundo argumento es el valor que necesita convertir.
- Si el argumento no es un número entero sin signo, vuestro programa deberá devolver 'Error' seguido de un salto de línea.
- Por razones de armonización, vuestro programa hablará en inglés.
- Vuestro programa debe analizar el diccionario dado como recurso para el proyecto. Los valores dentro de él deben usarse para imprimir el resultado. Estos valores podrán ser modificados.

- Cualquier memoria asignada en la heap (con malloc(3)) debe liberarse correctamente. Esto se verificará durante la evaluación.
- El diccionario seguirá las siguiente reglas:

```
[un numero][0 a n espacios]:[0 a n espacios][cualquier caracter imprimible]
```

  - Los números deben manejarse de la misma manera que atoi.
  - Debéis recortar los espacios al principio y al final del valor en el diccionario.
  - El diccionario siempre tendrá al menos las claves dadas en el diccionario de referencia. Su valor puede ser modificado; se pueden agregar entradas pero no se pueden eliminar las claves iniciales.
  - Solo tenéis que usar los valores dados inicialmente en el diccionario en anexos. (Por ejemplo, si añadimos la clave `54: fifty-four`, deberán utilizar siempre las mismas claves `50: fifty` y `4: four`)
  - Las entradas del diccionario pueden están ordenadas en cualquier orden.
  - Puede haber líneas vacías en el diccionario.
  - Si tenéis algún error al analizar el diccionario, vuestro programa debería devolver "Dict Error\n".
  - Si el diccionario no os permite solucionar el valor demandado, vuestro programa debería devolver "Dict Error\n".
- Ejemplo:

```
$> ./rush-02 42 | cat -e
forty two$
$> ./rush-02 0 | cat -e
zero$
$> ./rush-02 10.4 | cat -e
error$
$> ./rush-02 100000 | cat -e
one hundred thousand$
$> grep "20" numbers.dict | cat -e
20   :     hey     everybody !$
$> ./rush-02 20 | cat -e
hey everybody !$
```

Capítulo IV

Extras

- Utilizad `-`, `,`, `and` para estar más cerca de la sintaxis correcta.
- Haced el mismo ejercicio con un idoma diferente. Para ello se permite proporcionar otro diccionario que contenga las entradas necesarias.
- Cuando no haya argumento, utilzad `read` para leer la entrada estándar.

Capítulo V

Entrega y evaluación

Entrega tu proyecto en tu repositorio `Git` como de costumbre. Solo el trabajo entregado en el repositorio será evaluado durante la defensa. No dudes en comprobar varias veces los nombres de los archivos para verificar que sean correctos.

Como este proyecto no es comprobado por un programa, puedes organizar tus archivos como consideres oportuno, siempre y cuando entregues los archivos obligatorios y estos cumplan con los requisitos.

`Sólo necesitas entregar los archivos requeridos por el enunciado de este proyecto.`