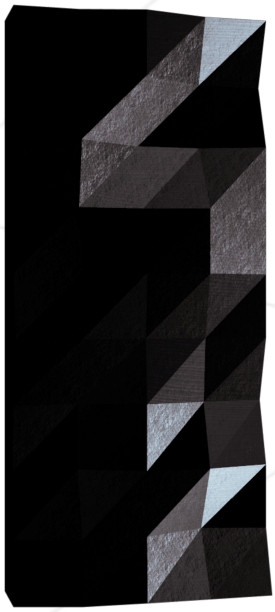

Piscina C

Rush 02

Sumário: Este documento é o enunciado do módulo C 02 da Piscina C da 42.

Versão: 10.3

Conteúdo

I	**Instruções**	**2**
II	**Preâmbulo**	**4**
III	**Tema**	**5**
IV	**Bônus**	**7**
V	**Entrega e avaliação entre pares**	**8**

Capítulo I

Instruções

- O grupo SERÁ inscrito para a avaliação. Automaticamente. Se algum de vocês cancelar a avaliação, não haverá outra.
- Qualquer pedido de esclarecimento sobre um dos temas provavelmente complicará o tema.
- Você deve seguir o procedimento de entrega para todos os seus exercícios.
- O tema pode mudar até uma hora antes da entrega.
- O programa deverá compilar com as flags: -Wall -Wextra -Werror, e utiliza `cc`.
- Se o seu programa não compila, você terá `0`.
- Seu programa deve ser escrito de acordo com a Norma. Se você tiver arquivos/-funções bônus, eles serão incluídos na verificação de norma e você receberá um 0 caso haja algum erro neles.
- Você deve entregar um Makefile, que compilará o seu projeto usando as regras `$NAME`, `clean` e `fclean`
- Você deve fazer o projeto com o time imposto a você e estar presente para a avaliação que for marcada, com todos os seus parceiros.
- Cada membro do grupo deve estar totalmente ciente do conteúdo do trabalho. Se vocês decidirem dividir o trabalho, assegure-se de que todos entendam o que todos fizeram. Durante a avaliação, você responderá perguntas e a nota final será baseada nas piores explicações.
- Reunir o grupo é de sua responsabilidade. Vocês têm todos os meios para entrar em contato com seus parceiros: telefone, email, pombo-correio, bola de cristal, etc. Nem precisa inventar desculpas. A vida não é justa, ela só é como é.
- Se, depois de realmente tentar de tudo, um de seus parceiros ainda estiver incomunicável: faça o seu rush e arranjaremos uma solução na defesa. Mesmo se foi o líder

do grupo: vocês todos possuem acesso ao repositório.

Capítulo II

Preâmbulo

A seguir, uma receita de uma tradicional torta de noz pecã:

Ingredientes
- Massa folhada
- 3/4 de manteiga sem sal
- 1 1/4 xícaras de açúcar mascavo claro
- 3/4 xícara de xarope de milho light
- 2 colheres de chá de extrato de baunilha puro
- 1/2 colher de chá de raspas de laranja
- 1/4 colher de chá de sal
- 3 ovos grandes
- 2 xícaras de metades de nozes (1/2 libra)

Acompanhamento: chantilly ou sorvete de baunilha

Preparação:
Pré-aqueça o forno a 350 ° F com uma assadeira na prateleira do meio.
Abra a massa em uma superfície levemente enfarinhada com um rolo levemente enfarinhad
Apare a borda, deixando uma saliência de 1,2 cm.
Dobre a saliência para baixo e pressione levemente contra a borda do prato de torta e
Pique levemente o fundo com um garfo.
Refrigere até ficar firme, pelo menos 30 minutos (ou congele 10 minutos).
Enquanto isso, derreta a manteiga em uma panela pequena e pesada em fogo médio.
Adicione o açúcar mascavo, mexendo até ficar homogêneo.
Retire do fogo e misture o xarope de milho, a baunilha, as raspas e o sal.
Bata levemente os ovos em uma tigela média e, em seguida, bata na mistura de xarope d
Coloque as nozes na casca da torta e despeje a mistura de xarope de milho uniformemen
Asse na assadeira quente até que o recheio esteja definido, 50 minutos a 1 hora.
Esfrie completamente.

Notas do cozinheiro:
A torta pode ser assada 1 dia antes e resfriada. Leve à temperatura ambiente antes de

Capítulo III

Tema

	Exercício : 00
rush-02	
Pasta de entrega : $ex00/$	
Ficheiros para entregar : `Makefile e todos os ficheiros necessários`	
Funções autorizadas : `write, malloc, free, open, read, close`	

- Escreva um programa que pegue um número como argumento e converta-o para sua forma escrita por extenso.
- Nome do executável: `rush-02`
- O código será compilado pelo comando:

```
make fclean
make
```

- O programa pode receber até 2 argumentos:
  - Se existir apenas um argumento, será o valor a converter;
  - Caso existam dois argumentos, o primeiro argumento será o novo dicionário de referência e o segundo argumento é o valor que você precisa converter.
- Se o argumento não for um inteiro positivo, o programa deve retornar "Error"seguido de uma quebra de linha.

`Seu programa deve ir além do unsigned int.`

- Seu programa deve parsear o dicionário dado como recurso ao projeto. Os valores contidos nele devem ser usados para imprimir o resultado. Esses valores podem ser modificados.

- Qualquer memória alocada na heap (com malloc(3)) deve ser liberada corretamente. Isso será verificado durante a avaliação.

- O dicionário seguirá estas regras:

```
[um numero][0 a n espacos]:[0 a n espacos][quaisquer caracteres imprimiveis]\n
```

  - Os números devem ser tratados da mesma forma que a função atoi os trata.
  - Os espaços antes e depois de um valor no dicionário devem ser suprimidos.
  - O dicionário deverá manter pelo menos as chaves como no dicionário de referência. Os valores podem ser modificados, mais entradas podem ser adicionadas, mas as chaves iniciais não podem ser removidas.
  - Você precisará usar somente as entradas iniciais (por exemplo, se for adicionado `54:  fifty-four`, você ainda deverá usar `50:  fifty` e `4:  four`).
  - As entradas do dicionário podem ser armazenadas em qualquer ordem.
  - Podem existir linhas vazias no dicionário.
  - No caso de erro no parseamento do dicionário, o programa deve retornar "Dict Error\n".
  - Se o dicionário não permite que voc6e resolva o valor pedido, o programa deve retornar "Dict Error\n".

- Exemplo:

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

Capítulo IV

Bônus

- Utilizar `,`, `-`, `and` para ficar mais próximo do que é sintaticamente correto.
- Fazer o mesmo exercício em outro idioma. Para isso, é permitido que voc6e forneça outro dicionário que contenha as entradas necessárias.
- Utilizar `read` para ler a partir do *standard input* quando não houver argumento.

Capítulo V

Entrega e avaliação entre pares

Entregue seu projeto em seu repositório `Git` como de costume. Somente o trabalho contido em seu repositório será avaliado durante a defesa. Não hesite em verificar mais de uma vez os nomes dos seus arquivos para ter certeza de que eles estão corretos.

Como esses exercícios não são verificados por um programa, fique à vontade para organizar seus arquivos como desejar, desde que você entregue os arquivos obrigatórios e preencha os requisitos do projeto.

`Você deve submeter somente os arquivos solicitados pelo subject deste projeto.`