



C 물고기 러쉬 02

요약: 이 문서는 C Piscine @ 42의 Rush02에 대한 주제입니다.

내용물

NS	지침	2
II	머리말	삼
III	주제	4
IV	보너스	6

1장

지침

- 그룹은 방어에 등록됩니다. 자동으로. 둘 중 한 사람이 취소하면 다른 사람은 없을 것입니다.
- 주제에 관한 모든 질문은 주제를 복잡하게 만들 것입니다.
- 모든 연습은 제출 절차를 따라야 합니다.
- 이 주제는 제출 1시간 전까지 변경될 수 있습니다.
- Moulinette는 다음 플래그를 사용하여 컴파일합니다. -Wall -Wextra -Werror; 그리고 용도gcc.
- 프로그램이 컴파일되지 않으면 0.
- \$ 규칙을 사용하여 프로젝트를 컴파일할 Makefile을 제출해야 합니다.NAME님, 깨끗한 그리고 fclean
- 따라서 당신은 부과된 팀과 함께 프로젝트를 수행하고 할당된 수비 슬롯에 모든 팀원과 함께 나타나야 합니다.
- 그룹의 각 구성원은 프로젝트의 작업을 완전히 알고 있어야 합니다. 워크로드를 분할하기로 선택한 경우 모든 사람이 수행한 작업을 모두 이해해야 합니다. 방어하는 동안 질문을 받게 되며 최종 점수는 최악의 설명을 기반으로 합니다.
- 그룹을 모으는 것은 귀하의 책임입니다. 전화, 이메일, 캐리어 비둘기, 영매술 등 팀원들과 연락할 수 있는 모든 수단이 있습니다. 그러니 굳이 핑계를 대지 마세요. 인생은 항상 공평하지 않습니다. 그냥 그렇습니다.
- 그러나 정말로 모든 것을 시도했다면 팀원 중 한 명이 도달할 수 없는 상태로 남아 있습니다. 어쨌든 프로젝트를 수행하면 수비 중에 이 문제에 대해 우리가 할 수 있는 것을 시도하고 볼 것입니다. 그룹 리더가 누락된 경우에도 제출 디렉토리에 액세스할 수 있습니다.
- 당연한 말이지만, 당신의 작업은 규범을 존중해야 합니다. 열심히 하세요. 그리고 즐기세요!

제2장

머리말

다음은 구식 피칸 파이 레시피입니다.

재료

- 페이스트리 반죽
- 3/4 스틱 무염 버터
- 1 1/4 컵 포장된 밝은 갈색 설탕
- 가벼운 옥수수 시럽 3/4컵
- 순수한 바닐라 추출물 2작은술
- 강판에 간 오렌지 제스트 1/2작은술
- 소금 1/4작은술
- 큰 달걀 3개
- 피칸 반쪽 2컵(1/2파운드)

준비물: 휘핑크림 또는 바닐라 아이스크림

준비:

중간 랙에 베이킹 시트를 놓고 오븐을 350°F로 예열합니다.

밀가루를 가볍게 뿌린 롤링 핀으로 가볍게 가루를 뿌린 표면에 반죽을 밀어 12인치 원형으로 만들고 9인치 파이 접시에 맞춥니다.

가장자리를 자르고 1/2인치 돌출부를 남깁니다.

오버행을 아래로 접고 파이 플레이트의 가장자리를 가볍게 누른 다음 장식적으로 압착합니다.

포크로 바닥 전체를 살짝 찌러줍니다.

단단해질 때까지 최소 30분(또는 10분 냉동)을 식힙니다. 그 사이에 중불에서 작고 무거운 스투 냄비에 버터를 녹입니다. 갈색 설탕을 넣고 부드러워질 때까지 휘젓습니다.


열에서 제거하고 옥수수 시럽, 바닐라, 제스트 및 소금을 털어냅니다. 중간 크기의 불에 계란을 가볍게 휘저은 다음 옥수수 시럽 혼합물을 휘젓습니다. 파이 껍질에 피칸을 넣고 콘 시럽 혼합물을 골고루 붓습니다. 채우기가 설정될 때까지 뜨거운 베이킹 시트에서 50분에서 1시간 굽습니다. 완전히 식히십시오.

쿡 노트:

파이는 하루 전에 구워서 식힐 수 있습니다. 서빙하기 전에 실온에 가져옵니다.

제3장

주제

	운동 00
	러시-02
	제출 디렉토리: 전00/제출할 파일: Makefile 및 필요한 모든 파일허용되는
	기능: 쓰기, malloc, 비어 있는, 열기, 읽기, 닫기

- 숫자를 인수로 받아 작성된 문자 값으로 변환하는 프로그램을 작성하십시오.
- 실행 파일 이름: 러시-02
- 소스 코드는 다음과 같이 컴파일됩니다.

```
fclean을 만들다
만들다
```

- 프로그램은 최대 2개의 인수를 사용할 수 있습니다.
 - 인수가 하나만 있는 경우 변환해야 하는 값입니다.
 - 두 개의 인수가 있는 경우 첫 번째 인수는 새 참조 사전이고 두 번째 인수는 변환해야 하는 값입니다.
- 인수가 유효한 부호 없는 정수가 아니면 프로그램은 "오류"를 반환하고 그 뒤에 줄 바꿈이 와야 합니다.
- 프로그램은 프로젝트에 리소스로 제공된 사전을 구문 분석해야 합니다. 내부의 값은 결과를 인쇄하는 데 사용해야 합니다. 이러한 값은 수정할 수 있습니다.
- 힙에 할당된 모든 메모리(malloc(3) 사용)는 올바르게 해제되어야 합니다. 이는 평가 중에 확인됩니다.
- 사전에는 다음과 같은 규칙이 있습니다.

```
[숫자][0 ~ n 공백]:[0 ~ n 공백][인쇄 가능한 모든 문자]\n
```

- 숫자는 atoi가 처리하는 것과 같은 방식으로 처리해야 합니다.
- 사전에서 값 앞뒤의 공백을 자릅니다.
- 사전에는 항상 참조 사전에서와 같이 최소한 키가 있습니다. 값을 수정할 수 있고 더 많은 항목을 추가할 수 있지만 초기 키는 제거할 수 없습니다.
- 초기 항목만 사용하면 됩니다(예: 54: 쉼넷,당신은 여전히 사용해야합니다 50: 오십 그리고 4: 4)
- 사전의 항목은 임의의 순서로 저장할 수 있습니다.
- 사전에는 빈 줄이 있을 수 있습니다.
- 사전 구문 분석에서 오류가 발생하면 프로그램에서 "Dict Error\n"을 출력해야 합니다.
- 사전에서 요청한 값을 해결할 수 없으면 프로그램에서 "Dict Error\n"을 출력해야 합니다.

• 예시:

```
$> ./rush-02 42 | 고양이 - 42$
$> ./rush-02 0 | 고양이 - 0$
$> ./rush-02 10.4 | 고양이 - 오류$
$> ./rush-02 100000 | 고양이 - 십만$

$> grep "20" number.dict | cat -e 20 : 안녕
하세요 여러분!$
$> ./rush-02 20 | 고양이 - 안녕
하세요 여러분!$
```

제4장

보너스

- -,,, 그리고 올바른 서면 구문에 더 가깝도록
- 다른 언어로 같은 운동을 합니다. 이를 위해 필요한 항목을 포함할 다른 사전을 제공할 수 있습니다.
- 사용 읽다 인수가 없을 때 표준 항목을 읽으려면