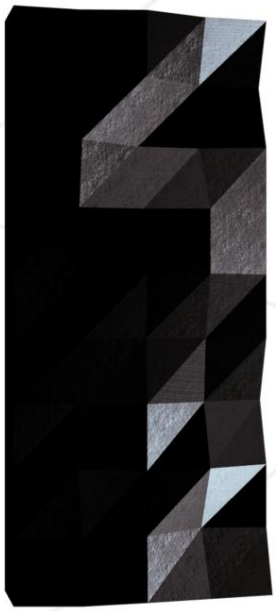

리프트

나만의 도서관

요약: 이 프
로젝트는 C 라이브러리 코딩에 관한 것입니다.
여기에는 프로그램이 의존할 범용 기능이 많이 포함됩니다.

버전: 15

내용물

1장

소개

C 프로그래밍은 매우 유용한 표준 기능에 액세스할 수 없을 때 매우 지루할 수 있습니다. 이 프로젝트는 이러한 기능이 작동하는 방식을 이해하고 구현하고 사용하는 방법을 배우는 것입니다. 당신의 의지가 당신만의 도서관을 만듭니다. 다음 C 학교 과제에서 사용할 것이기 때문에 도움이 될 것입니다.

일년 내내 libft를 확장하는 시간을 가지십시오. 그러나 새 프로젝트에서 작업할 때 라이브러리에서 사용되는 기능이 프로젝트 지침에서 허용되는지 확인하는 것을 잊지 마십시오.

제2장

공통 지침

- 프로젝트는 C로 작성해야 합니다.
- 프로젝트는 규범에 따라 작성되어야 합니다. 보너스 파일/기능이 있는 경우 표준 검사에 포함되며 내부에 표준 오류가 있으면 0이 표시됩니다.
- 정의되지 않은 동작을 제외하고 기능이 예기치 않게 종료되지 않아야 합니다(세그먼트 오류, 버스 오류, 이중 자유 등). 이 경우 프로젝트는 작동하지 않는 것으로 간주되고 평가 중에 0을 받습니다.
- 필요한 경우 모든 힙 할당 메모리 공간을 적절하게 해제해야 합니다. 누출 없음 용납됩니다.
- 주제가 요구하는 경우 -Wall, -Wextra 및 -Werror 플래그를 사용하여 소스 파일을 필수 출력으로 컴파일하고 cc를 사용하는 Makefile을 제출해야 하며 Makefile은 다시 연결되어서는 안 됩니다.
- Makefile은 최소한 $(NAME), all, clean, fclean 및 규칙을 포함해야 합니다. 답장.
- 프로젝트에 보너스를 제공하려면 Makefile에 규칙 보너스를 포함해야 합니다. 이 보너스는 프로젝트의 주요 부분에서 금지된 다양한 헤더, 라이브러리 또는 기능을 모두 추가합니다. 제목이 다른 것을 지정하지 않으면 보너스는 다른 파일 _bonus.{c/h}에 있어야 합니다. 필수 및 보너스 부분 평가는 별도로 수행됩니다.
- 프로젝트에서 libft를 사용할 수 있는 경우 해당 소스 및 관련 Makefile을 관련 Makefile과 함께 libft 폴더에 복사해야 합니다. 프로젝트의 Makefile은 해당 Makefile을 사용하여 라이브러리를 컴파일한 다음 프로젝트를 컴파일해야 합니다.
- 이 작업 은 제출할 필요가 없고 채점되지 않더라도 프로젝트에 대한 테스트 프로그램을 만들 것을 권장합니다 . 그것은 당신에게 당신의 작업과 동료의 작업을 쉽게 테스트 할 수있는 기회를 줄 것입니다. 이러한 테스트는 방어 중에 특히 유용하다는 것을 알게 될 것입니다. 실제로 방어하는 동안 자신의 테스트 및/또는 평가 중인 동료의 테스트를 자유롭게 사용할 수 있습니다.
- 작업을 할당된 git 저장소에 제출합니다. git 저장소에 있는 작업만 채점됩니다. 작업을 평가하기 위해 Deepthought가 할당되면 완료됩니다.

동료 평가 후. Deepthink의 채점 중 작업의 어느 부분에서 오류가 발생하면 평가가 중지됩니다.

제3장

필수 부분

프로그램 이름	libft.a
파일 제출	Makefile, libft.h, ft_*.c
Makefile 외	NAME, all, clean, fclean, re
부 함수.	아래에서 자세히
Libft 승인 설명	해당 사항 없음
	나만의 라이브러리 작성: 함수 모음 그것은 당신의 cursus에 유용한 도구가 될 것입니다.

III.1 기술적 고려사항

- 전역 변수 선언은 금지되어 있습니다.
- 더 복잡한 함수를 분할하기 위해 도우미 함수가 필요한 경우 정적 함수로 정의하십시오.
  기능. 이렇게 하면 해당 파일의 범위가 해당 파일로 제한됩니다.
- 모든 파일을 리포지토리의 루트에 배치합니다.
- 사용하지 않는 파일의 반입은 금지되어 있습니다.
- 모든 .c 파일은 -Wall -Wextra -Werror 플래그로 컴파일해야 합니다.
- 라이브러리를 생성하려면 ar 명령을 사용해야 합니다. libtool 명령 사용
  금지되어 있습니다.
- libft.a는 저장소의 루트에 생성되어야 합니다.

III.2 파트 1 - Libc 함수

시작하려면 libc에서 함수 집합을 다시 실행해야 합니다. 함수는 동일한 프로토타입을 가지며 원본과 동일한 동작을 구현합니다. 그들은 자신의 사람에게 정의된 방식을 따라야 합니다. 유일한 차이점은 이름뿐입니다. 'ft_' 접두사로 시작합니다. 예를 들어 strlen은 ft_strlen이 됩니다.

다시 실행해야 하는 함수의 프로토타입 중 일부는 '제한' 한정자를 사용합니다. 이 키워드는 c99 표준의 일부입니다. 따라서 자신의 프로토타입에 포함하고 -std=c99 플래그로 코드를 컴파일하는 것은 금지되어 있습니다.

다음과 같은 원래 기능을 구현하는 고유한 기능을 작성해야 합니다. 외부 기능이 필요하지 않습니다.

- 이알파
- isdigit
- 섬
- isascii
- 이스프린트
- strlen
- 멤셋
- 영
- memcpy
- 메모무브
- strlcpy
- strlcat
- 토퍼
- 낮추다
- strchr
- strrcr
- strncmp
- memchr
- memcmp
- strnstr
- 아토이

다음 두 함수를 구현하려면 malloc()을 사용합니다.

- 콜록
- strdup

III.3 파트 2 - 추가 기능

이 두 번째 부분에서는 libc에 없는 함수 집합을 개발해야 합니다.
또는 그것의 일부이지만 다른 형태입니다.

다음 기능 중 일부는 작성에 유용할 수 있습니다.
1부의 기능.

기능 이름	ft_substr
원기	char *ft_substr(char const *s, unsigned int start, size_t len);
파일 제출	-
매개변수	s: 하위 문자열을 생성할 문자열입니다. start: 하위 문자열의 시작 인덱스 문자열 ''. len: 부분 문자열의 최대 길이.
반환 값	하위 문자열입니다. 할당이 실패하면 NULL입니다.
외부 기능.	말록
설명	(malloc(3)을 사용하여) 할당하고 하위 문자열을 반환합니다. 문자열 ''에서. 하위 문자열은 인덱스 'start'에서 시작하며 다음 중 하나입니다. 최대 크기 'len'.

기능 이름	ft_strjoin
원기	char *ft_strjoin(char const *s1, char const *s2);
파일 제출	-
매개변수	s1: 접두사 문자열. s2: 접미사 문자열.
반환 값	새 문자열입니다. 할당이 실패하면 NULL입니다.
외부 기능.	말록
설명	(malloc(3)을 사용하여) 할당하고 새로운 값을 반환합니다. 연결의 결과인 문자열 '1'과 '2'입니다.

기능 이름	ft_strtrim
원기	char *ft_strtrim(char const *s1, char const *set);
파일 제출	-
매개변수	s1: 트리밍할 문자열. set: 트리밍할 문자의 참조 집합입니다.
반환 값	트리밍된 문자열입니다. 할당이 실패하면 NULL입니다.
외부 기능.	말록
설명	(malloc(3)을 사용하여) 할당하고 다음의 복사본을 반환합니다. 'set'에 지정된 문자가 제거된 '1' 문자열의 시작과 끝에서.

기능 이름	ft_split
원기	char **ft_split(char const *s, char c);
파일 제출	-
매개변수	s: 분할할 문자열입니다. c: 구분 문자.
반환 값	분할로 인한 새 문자열의 배열입니다. 할당이 실패하면 NULL입니다.
외부 기능.	malloc, 무료
설명	(malloc(3)을 사용하여) 할당하고 배열을 반환합니다. 다음을 사용하여 ''를 분할하여 얻은 문자열 문자 'c'를 구분 기호로 사용합니다. 배열이 끝나야 합니다. NULL 포인터로.

기능 이름	ft_itoa
원기	char *ft_itoa(int n);
파일 제출	-
매개변수	n: 변환할 정수.
반환 값	정수를 나타내는 문자열입니다. 할당이 실패하면 NULL입니다.
외부 기능.	말록
설명	(malloc(3)을 사용하여) 할당하고 문자열을 반환합니다. 인수로 받은 정수를 나타냅니다. 음수는 처리해야 합니다.

기능 이름	ft_strmapi
원기	char *ft_strmapi(char const *s, char(*f)(무부호 정수, 문자));
파일 제출	-
매개변수	s: 반복할 문자열입니다. f: 각 문자에 적용할 기능입니다.
반환 값	연속 애플리케이션에서 생성된 문자열 끄다'. 할당에 실패하면 NULL을 반환합니다.
외부 기능.	말록
설명	의 각 문자에 'f' 기능을 적용합니다. 문자열 '', 인덱스를 첫 번째 인수로 전달 새로운 문자열을 생성하려면(malloc(3) 사용) 결과 'f'의 연속 적용에서.

기능 이름	ft_striteri
원기	무효 ft_striteri(문자 *s, 무효(*f)(부호 없는 정수, 숯*));
파일 제출	-
매개변수	s: 반복할 문자열입니다. f: 각 문자에 적용할 기능입니다.
반환 값 외부 함수.	없음
	없음
설명	의 각 문자에 'f' 기능을 적용합니다. 인덱스를 전달하는 인수로 전달된 문자열 첫 번째 인수로. 각 문자는 다음을 통해 전달됩니다. 주소를 'f'로 변경하여 필요한 경우 수정합니다.

기능 이름	ft_putchar_fd
원기	무효 ft_putchar_fd(char c, int fd);
파일 제출	-
매개변수	c: 출력할 문자. fd: 쓸 파일 설명자.
반환 값 외부 함수.	없음
	쓰다
설명	지정된 파일에 문자 'c'를 출력합니다. 설명자.

기능 이름	ft_putstr_fd
원기	무효 ft_putstr_fd(char *s, int fd);
파일 제출	-
매개변수	s: 출력할 문자열입니다. fd: 쓸 파일 설명자.
반환 값 외부 함수.	없음
	쓰다
설명	주어진 파일에 문자열 's'를 출력 설명자.

기능 이름	ft_putendl_fd
원기	무효 ft_putendl_fd(char *s, int fd);
파일 제출	-
매개변수	s: 출력할 문자열입니다. fd: 쓸 파일 설명자.
반환 값	없음
외부 기능.	쓰다
설명	주어진 파일 디스크립터에 문자열 's'를 출력합니다. 그 뒤에 개행 문자가 옵니다.

기능 이름	ft_putnbr_fd
원기	무효 ft_putnbr_fd(int n, int fd);
파일 제출	-
매개변수	n: 출력할 정수입니다. fd: 쓸 파일 설명자.
반환 값	없음
외부 기능.	쓰다
설명	주어진 파일에 정수 'n'을 출력합니다. 설명자.

제4장

보너스 부분

필수 부분을 완료했다면 주저하지 말고 이 추가 부분을 수행하십시오. 성공적으로 통과하면 보너스 포인트가 제공됩니다.

메모리와 문자열을 조작하는 함수는 매우 유용합니다. 그러나 곧 알게 될 것입니다.
목록을 조작하는 것이 훨씬 더 유용합니다.

목록의 노드를 나타내려면 다음 구조를 사용해야 합니다. libft.h 파일에 선언을 추가합니다.

```
형식 정의 구조체 {          s_list

    무효 구                 *콘텐츠; *다음;
    조체 s_list             t_list;
}
```

t_list 구조체의 멤버는 다음과 같습니다.

- 내용: 노드에 포함된 데이터입니다. void *는 모든 종
  류의 데이터를 저장할 수 있습니다.
- next: 다음 노드의 주소 또는 다음 노드가 마지막 노드인 경우 NULL입니다.

Makefile에서 make 보너스 규칙을 추가하여 libft.a에 보너스 기능을 추가하십시오.

필수 부분이 PERFECT인 경우에만 보너스 부분이 평가됩니다. Perfect는 필수 부분이 완벽하게 수행되어 오작동 없이 작동함을 의미합니다. 모든 필수 요구 사항을 통과하지 못한 경우 보너스 부분은 전혀 평가되지 않습니다.

목록을 쉽게 사용하려면 다음 기능을 구현하십시오.

기능 이름	ft_lstnew
원기	t_list *ft_lstnew(무효 *내용);
파일 제출	-
매개변수	content: 노드를 생성할 내용입니다.
반환 값 외부 함수.	새 노드
	말록
설명	(malloc(3)으로) 할당하고 새 노드를 반환합니다. 멤버 변수 'content'는 다음으로 초기화됩니다. 매개변수 '내용'의 값입니다. 변수 'next'는 NULL로 초기화됩니다.

기능 이름	ft_lstadd_front
원기	무효 ft_lstadd_front(t_list **lst, t_list *new);
파일 제출	-
매개변수	lst: 첫 번째 링크에 대한 포인터 주소 목록. new: 노드에 대한 포인터의 주소 목록에 추가되었습니다.
반환 값 외부 함수.	없음
	없음
설명	목록의 시작 부분에 'new' 노드를 추가합니다.

기능 이름	ft_lstsize
원기	int ft_lstsize(t_list *lst);
파일 제출	-
매개변수	lst: 목록의 시작 부분입니다.
반환 값 외부 함수.	목록의 길이
	없음
설명	목록의 노드 수를 계산합니다.

기능 이름	ft_lstlast
원기	t_list *ft_lstlast(t_list *lst);
파일 제출	-
매개변수	lst: 목록의 시작 부분입니다.
반환 값	목록의 마지막 노드
외부 기능.	없음
설명	목록의 마지막 노드를 반환합니다.

기능 이름	ft_lstadd_back
원기	무효 ft_lstadd_back(t_list **lst, t_list *new);
파일 제출	-
매개변수	lst: 첫 번째 링크에 대한 포인터 주소 목록. new: 노드에 대한 포인터의 주소 목록에 추가되었습니다.
반환 값 외부 함수.	없음
	없음
설명	목록 끝에 노드 'new'를 추가합니다.

기능 이름	ft_lstdelone
원기	무효 ft_lstdelone(t_list *lst, 무효(*del)(무효 *));
파일 제출	-
매개변수	lst: 해제할 노드입니다. del: 삭제에 사용된 함수의 주소 내용.
반환 값	없음
외부 기능.	무료
설명	매개변수로 노드를 취하고 메모리를 해제합니다. 주어진 함수 'del'을 사용하는 노드의 내용 매개변수로 노드를 해제합니다. 의 기억 'next'는 해제하면 안 됩니다.

기능 이름	ft_lstclear
원기	무효 ft_lstclear(t_list **lst, 무효(*del)(무효 *));
파일 제출	-
매개변수	lst: 노드에 대한 포인터의 주소입니다. del: 삭제에 사용된 함수의 주소 노드의 내용.
반환 값	없음
외부 기능.	무료
설명	주어진 노드와 모든 노드를 삭제하고 해제합니다. 함수 'del'을 사용하여 해당 노드의 후계자 그리고 무료(3). 마지막으로 목록에 대한 포인터를 다음으로 설정해야 합니다. 없는.

기능 이름	ft_lstiter
원기	무효 ft_lstiter(t_list *lst, 무효(*f)(무효 *));
파일 제출	-
매개변수	lst: 노드에 대한 포인터의 주소입니다. f: 반복하는 데 사용되는 함수의 주소 목록.
반환 값 외부 함수.	없음
	없음
설명	목록 'lst'를 반복하고 함수를 적용합니다. 각 노드의 내용에 'f'.

기능 이름	ft_lstmap
원기	t_list *ft_lstmap(t_list *lst, 무효 *(*f)(무효 *), 무효(*del)(무효 *));
파일 제출	-
매개변수	lst: 노드에 대한 포인터의 주소입니다. f: 반복하는 데 사용되는 함수의 주소 목록. del: 삭제에 사용된 함수의 주소 필요한 경우 노드의 내용입니다.
반환 값	새 목록입니다. 할당이 실패하면 NULL입니다.
외부 기능.	malloc, 무료
설명	목록 'lst'를 반복하고 함수를 적용합니다. 각 노드의 내용에 'f'. 새로운 생성 의 연속 적용 결과 목록 함수 'f'. 'del' 함수는 다음을 수행하는 데 사용됩니다. 필요한 경우 노드의 내용을 삭제합니다.

제5장

제출 및 동료 평가

평소와 같이 Git 리포지토리에서 할당을 제출합니다. 방어 중에는 리포지토리 내부의 작업만 평가됩니다. 파일 이름이 올바른지 다시 한 번 확인하는 것을 망설이지 마십시오.

모든 파일을 저장소의 루트에 배치합니다.