

C Piscine C 12

Summary: Bu döküman, 42'nin C Piscine'sinin C 12 modülünün dersidir.

# Contents

I	Önsöz	2
II	Yönergeler	4
III	Çalışma 00 : ft_create_elem	6
IV	Çalışma 01 : ft_list_push_front	7
$\mathbf{V}$	Çalışma 02 : ft_list_size	8
VI	Exercice 03 : ft_list_last	9
VII	Çalışma 04 : ft_list_push_back	10
VIII	Çalışma 05 : ft_list_push_strs	11
IX	Çalışma 06 : ft_list_clear	12
$\mathbf{X}$	Çalışma 07 : ft_list_at	13
XI	Çalışma 08 : ft_list_reverse	14
XII	Çalışma 09 : ft_list_foreach	15
XIII	Çalışma 10 : ft_list_foreach_if	16
XIV	Çalışma 11 : ft_list_find	17
XV	Çalışma 12 : ft_list_remove_if	18
XVI	Çalışma 13 : ft_list_merge	19
XVII	Çalışma 14 : ft_list_sort	20
XVIII	Çalışma 15 : ft_list_reverse_fun	21
XIX	Çalışma 16 : ft_sorted_list_insert	22
XX	Çalışma 17 : ft_sorted_list_merge	23

Chapter I Önsöz

> İZLEME KEYFİNİZ KAÇABİLİR BİR SONRAKİ SAYFAYI OKUMAYIN

### Uyarıldınız.

- In Yıldız Savaşları filmlerinde Darth Vader, Luke'un babası.
- $\bullet$  In Olağan Şüpheliler filminde Verbal, aslında Keyser Söze.
- In Dövüş Kulübü filminde Tyler Durden ve hikayeyi anlatan aslında aynı kişi.
- In Altıncı His filminde Bruce Willis filmin başından itibaren aslında ölü idi.
- In Diğerleri filminde evin yerleşik halkı hayaletler.
- In Bambi filminde Bambi'nin annesi ölüyor.
- In Köy filminde köylüler canavar ve film bizim zamanımızda geçiyor.
- In Harry Potter'da Dumbledore ölüyor.
- In Maymunlar Cehennemi filmi gezegenimizde geçiyor.
- In Taht Oyunları dizisinde Robb Stark ve Joffrey Baratheon düğün günlerinde ölüyorlar.
- In Alacakaranlık film serisinde vampirler güneş ışığı altında parlıyor.
- In Stargate SG-1 dizisinin ilk sezonunun 18. bölümünde, O'Neill ve Carter Antarktika'dalar.
- In Kara Şövalye Yükseliyor filminde, Miranda Tate aslında Talia Al'Ghul.
- In Super Mario Kardeşler'de prenses başka bir kalede.

#### Chapter II

### Yönergeler

- Lütfen sadece bu sayfayı referans alınız: söylentilere kulak asmayınız.
- Dikkat! Dokümanın gönderim öncesinde değişme ihtimali vardır.
- Lütfen dosyalarınız ve dizileriniz için gerekli yetkilere sahip olduğunuzdan emin olunuz.
- Bütün çalışmalarınız için gönderim talimatlarını takip ediniz.
- Çalışmalarınız sınıf arkadaşlarınız tarafından kontrol edilip notlandırılacaktır.
- Aynı zamanda, çalışmalarınız Moulinette adlı program tarafından da kontrol edilip notlandırılacaktır.
- Moulinette değerlendirmelerinde çok titiz ve katıdır. Otomatik bir program olmasından dolayı görüş alışverişi mümkün değildir. Sürpriz bir sonuçla karşılaşmamak için çalışmalarınızı dikkatlice yapınız.
- Moulinette çok açık görüşlü değildir. Kodunuz Norm'a uymadığı takdirde onu anlamaya çalışmayacaktır. Moulinette dosyalarınızın norm'a uyup uymadığını kontrol etmek için norminette adında bir program kullanmaktadır. TL;DR: norminette'in kontrolünden geçemeyecek bir dosya teslim etmek akılsızca olacaktır.
- Çalışmalar en kolaydan en zora olacak şekilde zorluklarına göre sıralanmıştır. Daha zor bir çalışma başarıyla tamamlanmış bile olsa daha kolay bir çalışmanın tamamıyla fonksiyonel olmaması durumunda dikkate alınmayacaktır.
- Yasaklanmış bir fonksiyon kullanmak hile olarak görülmektedir. Bunu yapan kişiler
   -42 puan alacaktır, ve bu not pazarlığa tabi değildir.
- Sizden program istersek sadece bir main() fonksiyonu göndermeniz gerekir.
- Moulinette çalışmaları şu şekilde sınıflandırır: -Wall -Wextra -Werror ve gcc
- Eğer programınız sınıflandırılamazsa, 0 alırsınız.
- Dizininizde konunun başlığındakiler dışında hiçbir dosya bırakmayınız.
- Bir sorunuz mu var? Sağınızdaki arkadaşınıza sorun. Olmadı solunuzdakine...

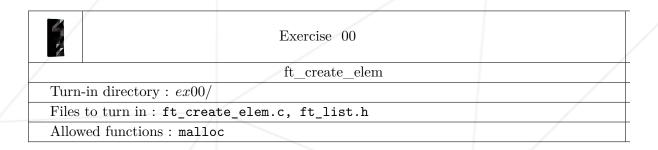
C Piscine

- ullet Başvuru kılavuzunuzun adı Google / man / the Internet / ... ' dır.
- Intranetteki forumun "C Piscine" kısmını ya da Slack'deki Piscine bölümünü kontrol edin.
- Konu içerisinde net bir şekilde belirtilmemiş detayları anlayabilmek için örnekleri dikkatlice inceleyiniz.
- Odin ve Thor adına! Kafayı çalıştırın!!!
- Bu dersin çalışmaları için aşağıdaki yapıyı kullanmanız gerekmektedir :

- Bu yapıyı bir ft\_list.h dosyasına koyup, her çalışma için yollamanız gerekmektedir.
- Çalışma 01'den itibaren ft\_create\_elem'imizi kullanacağız, buna göre önleminizi alın. (Prototipini ft\_list.h dosyasında bulundurmak işe yarayabilir...).

### Chapter III

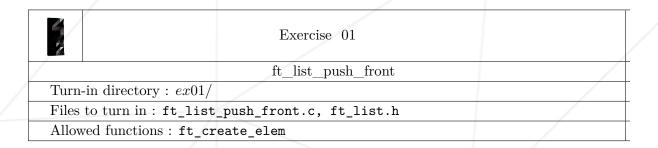
## Çalışma 00 : ft\_create\_elem



- t\_list tipinde yeni bir öğe yaratacak bir ft\_create\_elem fonksiyonu oluşturun.
- It should assign to the given argument and next to NULL. Verilmiş olan değişkene data ve NULL'a da next atamalıdır.
- Prototipi şu şekilde olmalıdır :

### Chapter IV

## Çalışma 01 : ft\_list\_push\_front

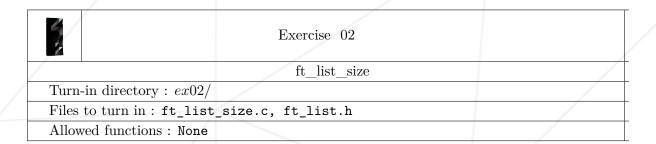


- Liste başına t\_list tipinde yeni bir öge ekleyen ft\_list\_push\_front oluşturun.
- Değişkene data atamalıdır.
- Eğer gerekli ise, liste başındaki işaretleyiciyi güncelleyecektir.
- Prototipi şu şekilde olmalıdır :

void ft\_list\_push\_front(t\_list \*\*begin\_list, void \*data);

## Chapter V

# Çalışma 02 : ft\_list\_size

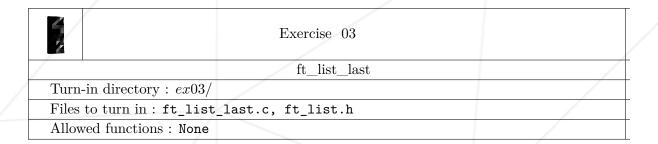


- Listedeki öğelerin sayısını geri bildirecek olan bir ft\_list\_size fonksiyonu oluşturun.
- Prototipi şu şekilde olmalıdır :

int ft\_list\_size(t\_list \*begin\_list);

## Chapter VI

Exercice 03: ft\_list\_last

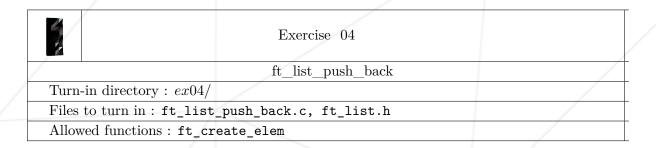


- Listedeki son öğeyi geri bildirecek olan bir ft\_list\_last fonksiyonu oluşturun.
- Prototipi şu şekilde olmalıdır :

t\_list \*ft\_list\_last(t\_list \*begin\_list);

### Chapter VII

## Çalışma 04 : ft\_list\_push\_back

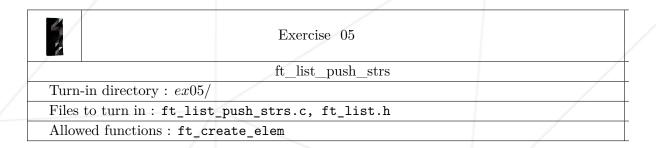


- Liste sonuna t\_list tipinde bir öge ekleyecek olan bir ft\_list\_push\_back fonksiyonu oluşturun.
- Verilmiş olan değişkene data atamalıdır.
- Eğer gerekli ise, liste başındaki işaretleyiciyi güncelleyecektir.
- Prototipi şu şekilde olmalıdır :

void ft\_list\_push\_back(t\_list \*\*begin\_list, void \*data);

## Chapter VIII

## Çalışma 05 : ft\_list\_push\_strs

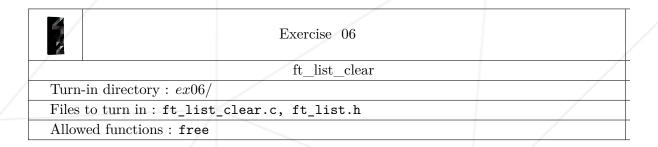


- strs öğesinde yer alan öğe tarafından gösterilen bütün diziyi içeren bir ft\_list\_push\_strs fonksiyonu oluşturun.
- size, strs'nin boyutudur.
- The first element should be at the end of the list. İlk öğe, listenin sonunda yer almalıdır.
- İlk bağlantının adresi listede geri döner.
- Prototipi şu şekilde olmalıdır :

t\_list \*ft\_list\_push\_strs(int size, char \*\*strs);

### Chapter IX

## Çalışma 06 : ft\_list\_clear

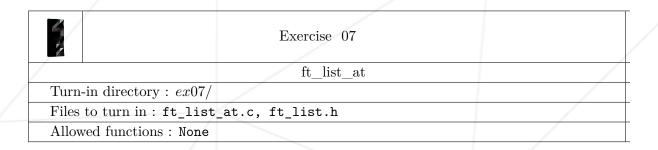


- Listeden bütün bağlantıları çıkartacak ve serbest bırakacak olan bir ft\_list\_clear fonksiyonu oluşturun.
- Her bir data'yı serbest bırakmak için free\_fct kullanılır.
- Prototipi şu şekilde olmalıdır :

void ft\_list\_clear(t\_list \*begin\_list, void (\*free\_fct)(void \*));

### Chapter X

## Çalışma 07 : ft\_list\_at



- nbr 0'a eşit olduğunda listenin ilk öğesi olduğunun bilindiği durumda, listenin N'inci öğesinin geri dönüşünü yapacak olan bir ft\_list\_at fonksiyonu oluşturun.
- Bir hata durumunda, geri dönüş boş bir işaretçi olmalıdır.
- Prototipi şu şekilde olmalıdır :

t\_list \*ft\_list\_at(t\_list \*begin\_list, unsigned int nbr);

## Chapter XI

# Çalışma 08 : ft\_list\_reverse

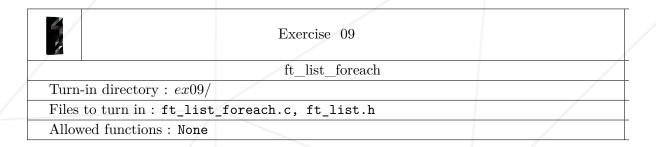
	Exercise 08	
/	ft_list_reverse	
Turn-in directory : $ex08$		
Files to turn in : ft_li		
Allowed functions: Non		

- Listedeki öğelerin sırasını tersine çevirecek bir ft\_list\_reverse fonksiyonu oluşturun. Bütün öğelerin değeri aynı kalmalıdır.
- $\bullet$ Fonksiyon bizim kendi ft\_list.h'mizi kullacaktır. Dikkat edin.
- Prototipi şu şekilde olmalıdır :

void ft\_list\_reverse(t\_list \*\*begin\_list);

### Chapter XII

## Çalışma 09 : ft\_list\_foreach



- Listenin her öğesine değişken olarak verilen fonksiyonu uygulayan bir ft\_list\_foreach fonksiyonu oluşturun.
- f listedeki aynı sırası ile uygulanmalıdır.
- Prototipi şu şekilde olmalıdır :

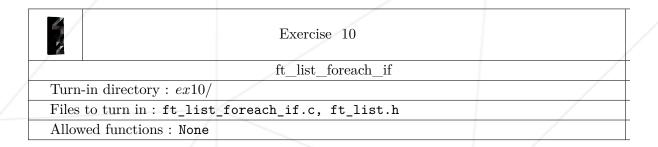
```
void ft_list_foreach(t_list *begin_list, void (*f)(void *));
```

• f tarafından gösterilen fonksiyon şu şekilde kullanılacaktır :

(\*f)(list\_ptr->data);

## Chapter XIII

## Çalışma 10 : ft\_list\_foreach\_if



- Listenin bazı öğelerine değişken olarak verilen fonksiyonu uygulayan bir ft\_list\_foreach\_if fonksiyon oluşturun.
- Fonksiyonu sadece cmp'nin data\_ref ile beraber olduğu öğelere uygulayın, cmp'nin geri dönüsü 0 olacaktır.
- f listedeki aynı sırası ile uygulanmalıdır.
- Prototipi şu şekilde olmalıdır :

```
void ft_list_foreach_if(t_list *begin_list, void (*f)(void *), void
*data_ref, int (*cmp)())
```

• f ve cmp tarafından gösterilen fonksiyonlar şu şekilde kullanılacaktır:

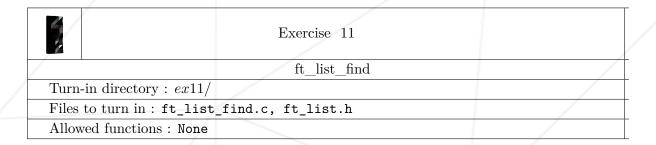
```
(*f)(list_ptr->data);
(*cmp)(list_ptr->data, data_ref);
```



Örneğin, cmp fonksiyonu, ft\_strcmp olabilir...

### Chapter XIV

# Çalışma 11 : ft\_list\_find



- cmp ile data\_ref'e kıyasla ilk öğenin verisinin adresinin 0'a geri dönüşünü sağlayacak olan bir ft\_list\_find fonksiyonu oluşturun.
- Prototipi şu şekilde olmalıdır :

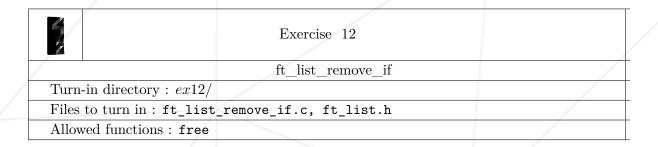
```
t_list *ft_list_find(t_list *begin_list, void *data_ref, int (*cmp)());
```

• cmp ile gösterilen fonksiyon şu şekilde kullanılacaktır :

(\*cmp)(list\_ptr->data, data\_ref);

### Chapter XV

# Çalışma 12 : ft\_list\_remove\_if



- cmp kullanan ve data\_ref ile karşılaştırıldığında geri dönüşü 0 olan bütün verileri listeden çıkartan bir ft\_list\_remove\_if fonksiyonu oluşturun.
- Silinecek olan bir öğeden veri free\_fct kullanarak serbest bırakılmalıdır.
- Prototipi şu şekilde olmalıdır :

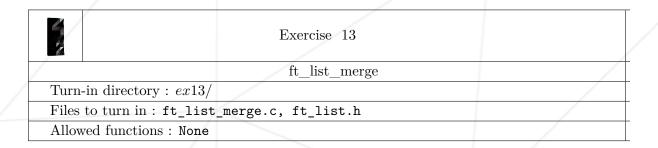
```
void ft_list_remove_if(t_list **begin_list, void *data_ref, int (*cmp)(), void (*free_fct)(void *))
```

• cmp ve free fct tarafından gösterilen fonksiyon şu şekilde kullanılacaktır :

```
(*cmp)(list_ptr->data, data_ref);
(*free_fct)(list_ptr->data);
```

## Chapter XVI

## Çalışma 13 : ft\_list\_merge

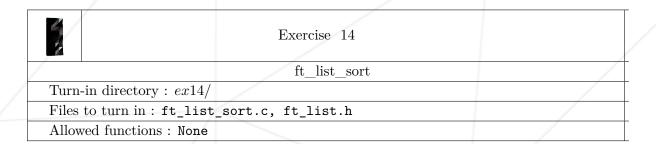


- begin2 listesinin öğelerini, begin1 listesinin sonuna koyacak bir ft\_list\_merge fonksiyonu oluşturun.
- Öğe oluşturmaya izin yoktur.
- Prototipi şu şekilde olmalıdır :

void ft\_list\_merge(t\_list \*\*begin\_list1, t\_list \*begin\_list2);

## Chapter XVII

## Çalışma 14 : ft\_list\_sort



- Öğelerin verisini bir fonksiyon kullanıp karşılaştırarak, listenin öğelerini küçükten büyüğe doğru dizen bir ft\_list\_sort fonksiyonu oluşturun.
- Prototipi şu şekilde olmalıdır :

```
void ft_list_sort(t_list **begin_list, int (*cmp)());
```

 $\bullet\,$ cmp tarafından gösterilen fonksiyon şu şekilde kullanıla<br/>caktır :

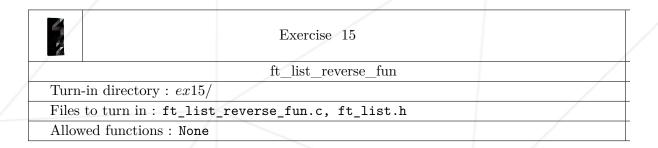
```
(*cmp)(list_ptr->data, list_other_ptr->data);
```



Örneğin, cmp, ft\_strcmp olabilir.

# Chapter XVIII

Çalışma 15 : ft\_list\_reverse\_fun

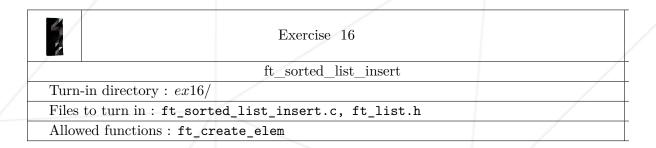


- Listedeki öğelerin sırasını tersine çeviren bir ft\_list\_reverse\_fun fonksiyonu oluşturun.
- Prototipi şu şekilde olabilir :

void ft\_list\_reverse\_fun(t\_list \*begin\_list);

### Chapter XIX

## Çalışma 16 : ft\_sorted\_list\_insert



- Yeni bir öge oluşturup, bu öğeyi listeye küçükten büyüğe sıralamasını bozmayacak şekilde yerleştiren bir ft\_sorted\_list\_insert fonksiyonu oluşturun.
- Prototipi şu şekilde olmalıdır :

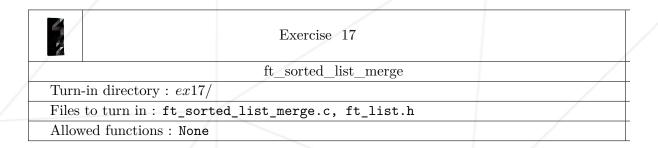
```
void ft_sorted_list_insert(t_list **begin_list, void *data, int (*cmp)());
```

• cmp tarafından gösterilen fonksiyon şu şekilde kullanılacaktır :

(\*cmp)(list\_ptr->data, list\_other\_ptr->data);

### Chapter XX

## Çalışma 17 : ft\_sorted\_list\_merge



- begin1 listesinin küçükten büyüğe sıralamasını koruyarak begin2 listesinin öğelerini ona entegre edecek olan bir ft\_sorted\_list\_merge fonksiyonu oluşturun.
- Prototipi şu şekilde olmalıdır :

```
void ft_sorted_list_merge(t_list **begin_list1, t_list *begin_list2, int (*cmp)());
```

 $\bullet\,$ cmp tarafından gösterilen fonksiyon şu şekilde kullanıla<br/>caktır :

(\*cmp)(list\_ptr->data, list\_other\_ptr->data);