

(https://profile.intra.42.fr)

## CPP MODÜLÜ 01 (/PROJECTS/CPP-MOD PROJESİ İÇİN ÖLÇEK (/PROJECTS/CPP-MC

Bu takımdaki 1 öğrenciyi değerlendirmelisiniz.



Git deposu

git@vogosphere.42istanbul.com.tr:vogosphere/intra-uuid-c9526a

### giriş

Lütfen aşağıdaki kurallara uyun:

- Değerlendirme süreci boyunca kibar, saygılı ve yapıcı olun . Topluluğun iyiliği buna bağlıdır.

- Çalışması değerlendirilen öğrenci veya grupla projelerindeki olası aksaklıkları belirleyin.

Belirlenen sorunları tartışmak ve değerlendirmek için zaman ayırın . Akranlarınızın projenin işlevselliğinin kapsamını

nasıl anladıkları konusunda bazı farklılıklar olabileceğini göz önünde bulundurmalısınız . Her ve onları olabildiğince dürüst bir şekilde değerlendirin . Pedagoji, ancak ve ancak akran değer yapıldığında faydalıdır.

### Yönergeler

- Yalnızca değerlendirilen öğrenci veya grubun Git deposunda teslim edilen çalışmayı notland

- Git deposunun öğrenci(ler)e ait olduğundan emin olun.

Projenin beklenen proje olduğundan emin olun. Ayrıca, 'git clone' komutunun

boş bir klasörde kullanıldığından emin olun. - Sizi kandırmak ve resmi deponun içeriği olmaya değerlendirmenizi

sağlamak için kötü amaçlı takma adların kullanılmadığından emin olun . - Herhangi bir sürpriz mümkünse, notlandırmayı kolaylaştırmak için kullanılan tüm komut dosyalarını (test veya otomasyon dosyaları) birlikte gözden geçirin. - Değerlendireceğiniz ödevi tamamlamadıysanız, değerlendirilmeden önce konunun tamamını okumalısınız. - Boş bir depo, çalışmayan bir program, N durumları bildirmek için mevcut bayrakları kullanın . Bu durumlarda, değerlendirme işlemi son veya hile durumunda -42 olur. Ancak, kopya çekme durumu hariç, öğrencilerin gelecekte tekrar hataları belirlemek için teslim edilen çalışmaları birlikte gözden geçirmeleri şiddetle tavsiye ed dosyası (varsa) dışında hiçbir dosyayı düzenlememelisiniz . Bir dosyayı düzenlemek istiyorsanız öğrenciyle nedenlerini açıkça belirtin ve ikinizin de bu konuda hemfikir olduğundan emin olun. sızıntılarının olmadığını da doğrulamalısınız. Yığın üzerinde ayrılan tüm bellek, yürütmenin so bir şekilde serbest bırakılmalıdır. Bilgisayarda bulunan farklı araçlardan herhangi birini (örneğin e\_fence) kullanabilirsiniz . Bellek sızıntısı durumunda, ilgili bayrağı işaretleyin.

 konu.pdf (https://cdn.intra.42.fr/pdf/pdf/197133/en.subject.pdf)

## Ön testler

*Hile yapıldığından şüpheleniliyorsa, değerlendirme burada durdurulur. Hile bildirimi için "Hile" kararı sakın ve akıllıca verin ve lütfen bu düğmeyi dikkatli kullanın.*

### Önkoşullar

Kodun C++ ile ve -Wall, -Wextra, -Werror bayraklarıyla derlenmesi gerekiyor. Bu projenin C++98 standardına uyması gerektiğini unutmayın. Bu nedenle, C++11 (ve sonrası) fonksiyonları veya konteynerleri BEKLENMEMEKTEDİR.

Bunlardan herhangi biri, söz konusu alıştırmayı notlandırmamanız gerektiği anlamına gelir:

- Bir fonksiyon, (şablon fonksiyonları hariç) bir başlık dosyasında uygulanır.
- Bir Makefile, gerekli bayraklar olmadan ve/veya C++'tan farklı bir derleyiciyle derleme y

Bunlardan herhangi biri, projeyi "Yasaklanmış

Fonksiyon" olarak işaretlemeniz gerektiği anlamına gelir:

- "C" fonksiyonlarının kullanımı (\*alloc, \*printf, free).
- Egzersiz yönergelerinde izin verilmeyen bir işlevin kullanılması.
- "using namespace <ns\_name>" veya "friend" anahtar kelimesinin kullanımı.
- Use of an external library, or features from versions other than C++98.

 Yes

 No

## Exercise 00: BraiiiiiiinnnzzzzZ

*The goal of this exercise is to understand how to allocate memory in C++.*

### Makefile and tests

There is a Makefile that compiles using the appropriate flags.

There is at least a main to test the exercise.

If this is not the case, the evaluation of this exercise stops here. you can move on to the next exercise.

 Yes

 No

### Zombie Class

There is a Zombie Class.

It has a private name attribute.

It has at least a constructor.

It has a member function announce( void ) that prints: "<name>: BraiiiiiiinnnzzzzZ..."

The destructor prints a debug message that includes the name of the zombie.

 Yes

 No

### newZombie

There is a newZombie() function prototyped as: [ Zombie\* newZombie( std::string name ); ]

It should allocate a Zombie on the heap and return it.

Ideally, it should call the constructor that takes a string and initializes the name.

The exercise should be marked as correct if the Zombie can announce itself with the name passed to the function.

There are tests to prove everything works.

The zombie is deleted correctly before the end of the program.

 Yes

 No

### randomChump

There is a randomChump() function prototyped as: [ void randomChump( std::string name ); ]  
It should create a Zombie on the stack, and make it announce itself.  
Ideally the zombie should be allocated on the stack (so implicitly deleted at the end of the function). It can also be allocated on the heap and then explicitly deleted.  
The student must justify their choices.  
There are tests to prove everything works.

✓ Yes

✗ No

## Exercise 01: Moar brainz!

*The goal of this exercise is to allocate a number of objects at the same time using new[], initially properly delete them.*

### Makefile and tests

There is a Makefile that compiles using the appropriate flags.  
There is at least a main to test the exercise.  
If this is not the case, the evaluation of this exercise stops here. you can move on to the next exercise.

✓ Yes

✗ No

### zombieHorde

The Zombie Class has a default constructor.  
There is a zombieHorde() function prototyped as: [ Zombie\* zombieHorde( int N, std::string name ); ]  
It allocates N zombies on the heap explicitly using new[].  
After the allocation, there is an initialization of the objects to set their name.  
It returns a pointer to the first zombie.  
There are enough tests in the main to prove the previous points.  
Like: calling announce() on all the zombies.  
Last, all the zombies should be deleted at the same time in the main.

✓ Yes

✗ No

## Exercise 02: HI THIS IS BRAIN

*Demystify references! Demystify references! Demystify references! Demystify references! Demystify references! Demystify references! Demystify references! Demystify references! Demystify references! Demystify references! Demystify references!*

### Makefile and tests

There is a Makefile that compiles using the appropriate flags.  
There is at least a main to test the exercise.  
If this is not the case, the evaluation of this exercise stops here. you can move on to the next exercise.

✓ Yes

✗ No

### HI THIS IS BRAIN

There is a string containing "HI THIS IS BRAIN".  
stringPTR is a pointer to the string.  
stringREF is a reference to the string.  
The address of the string is displayed using the string variable, the stringPTR and the stringREF.  
The variable content is displayed using the stringPTR and the stringREF.

✓ Yes

✗ No

## Exercise 03: Unnecessary violence

*The objective of this exercise is to understand that pointers and references present some similarities and make them less or more appropriate depending on the use and the lifecycle of the object used.*

### Makefile and tests

There is a Makefile that compiles using the appropriate flags.  
There is at least a main to test the exercise.  
If this is not the case, the evaluation of this exercise stops here. you can move on to the next exercise.

☒ Yes☐ No

### Weapon

There is a Weapon class that has a type string, a getType() and a setType().  
The getType() function returns a const reference to the type string.

☒ Yes☐ No

### HumanA and HumanB

HumanA can have a reference or a pointer to the Weapon.  
Ideally, it should be implemented as a reference, since the Weapon exists from creation until destruction, and never changes.  
HumanB must have a pointer to a Weapon since the field is not set at creation time, and the weapon can be NULL.

☒ Yes☐ No

## Exercise 04: Sed is for losers

*Thanks to this exercise, the student should have gotten familiar with ifstream and ofstream.*

### Makefile and tests

There is a Makefile that compiles using the appropriate flags.  
There is at least a main to test the exercise.  
If this is not the case, the evaluation of this exercise stops here. you can move on to the next exercise.

☒ Yes☐ No

### Exercise 04

There is a function replace (or other name) that works as specified in the subject.  
The error management is efficient: try to pass a file that does not exist, change the permissions, pass it empty, etc.  
If you can find an error that isn't handled, and isn't completely esoteric, no points for this exercise.  
The program must read from the file using an ifstream or equivalent, and write using an ofstream or equivalent.  
The implementation of the function should be done using functions from std::string, not by reading the string character by character.  
This is not C anymore!

☒ Yes☐ No

## Exercise 05: Harl 2.0

*The goal of this exercise is to use pointers to class member functions. Also, this is the opportunity to practice different log levels.*

### Makefile and tests

There is a Makefile that compiles using the appropriate flags.  
There is at least a main to test the exercise.  
If this is not the case, the evaluation of this exercise stops here. you can move on to the next exercise.

☒ Yes☐ No

### Our beloved Harl

Konuda gerekli olan en az 5 fonksiyona sahip Harl adında bir sınıf bulunmaktadır.  
`complain()` fonksiyonu, diğer fonksiyonları onlara işaretçi kullanarak çalıştırır.  
İdeal olarak, öğrencinin log seviyesine karşılık gelen farklı dizeleri ilgili üye fonksiyonlarının işaretçileriyle eşleştirmenir uygulaması gerekirdi.  
Uygulama farklı olsa bile alıştırma çalışıyorsa, geçerli olarak işaretlemelisiniz. İzin verilmeyen tek şey, çirkin bir `if/elseif/else` yapısı kullanma görüntülediği mesajı değiştirmeyi veya konuda verilen örnekleri görüntülemeyi seçebilirdi , her ikisi de geçerlidir.

[↩ Evet](#)[✕ HAYIR](#)

## Alıştırma 06: Harl filtresi

Artık deneyimli kod yazarları olduğunuza göre, yeni komut türlerini, ifadeleri, döngüleri vb. kull alıştırmanın amacı, switch ifadesini keşfetmenizi sağlamaktır.

### Makefile ve testler

Uygun bayraklar kullanılarak derleme yapan bir Makefile mevcut.

Egzersiz test etmek için en az bir main fonksiyonu bulunuyor.

Eğer durum böyle değilse, bu egzersizin değerlendirilmesi burada durur ve bir sonraki egzersi

[↩ Evet](#)[✕ HAYIR](#)

### Harl'ı Kapatma

`harlFilter` programı, argüman olarak "DEBUG", "INFO", "WARNING" veya "ERROR" gibi log herhangi birini alır . Ardından, yalnızca aynı seviyede veya daha üst seviyedeki (DEBUG < INI ERROR)

mesajları görüntülemelidir . Bu , varsayılan bir durum içeren bir `switch` ifadesi kullanılarak uy ediyorum, lütfen artık `if/elseif/else` kullanmayın.

[↩ Evet](#)[✕ HAYIR](#)

## Değerlendirmeler

Savunmaya karşılık gelen bayrağı kontrol etmeyi unutmayın.

[✓ Tamam](#)[★ Olağanüstü proje](#)[Boş iş](#)[📌 Tamamlanmamış iş](#)[📁 Aldatmak](#)[🚧 Kaza](#)[▲ Endişe verici durum](#)[🚫 Yasaklanmış işlev](#)[💬 Kodu destekleyemiyorum/açıklayar](#)

## Çözüm

Bu değerlendirmeye yorum bırakın (en fazla 2048 karakter)

[Değerlendirmeyi bitir](#)

[API Genel Kullanım Şartları](https://profile.intra.42.fr/legal/terms/33)  
(https://profile.intra.42.fr/legal/terms/33)

[Çerezlerin kullanımına ilişkin bildirim](https://profile.intra.42.fr/legal/terms/2)  
(https://profile.intra.42.fr/legal/terms/2)

[Gizlilik politikası](https://profile.intra.42.fr/legal/terms/5)  
(https://profile.intra.42.fr/legal/terms/5)

[Sitenin genel kullanım şartları](https://profile.intra.42.fr/legal/terms/6)  
(https://profile.intra.42.fr/legal/terms/6)

[42 Türk](https://profile.ir)  
(https://profile.ir)