

## Q0:

```

0-isupper
=====
A function that checks for uppercase character.

• Prototype: int _isupper(int c);
• Returns: 1 if c is uppercase
• Returns: 0 otherwise

FILE: The standard library provides a similar function: isupper, but see isupper(3) for more.

#include <cs50.h>
#include "main.h"
#include <ctype.h>
#include <cs50.h>

int _isupper(int c)
{
    // make c lowercase
    if (c >= 'A' && c <= 'Z')
        return 1;
    else
        return 0;
}

int main(void)
{
    // ask user for character
    char c = getchar();
    printf("Is %c uppercase? %d\n", c, _isupper(c));
    printf("Is %c uppercase? %d\n", c, isupper(c));
    return 0;
}

```

## المطلوب

تكتب دالة اسمها **\_isupper** ، وظيفتها:

- تأخذ حرف **c** (القيمة الحرفية في C تخزن كأرقام ASCII).
- ترجع 1 إذا كان الحرف كبير (A-Z).
- ترجع 0 إذا كان غير كبير (a-z) أو أي شيء ثانٍ.

## ملاحظات

- فيه دالة جاهزة في مكتبة C القياسية اسمها **isupper()**، بس هنا نبغى نسوى وحدتنا الخاصة.
- نحتاج نكتب الكود داخل ملف اسمه **0-isupper.c**.
- فيه ملف **main.h** لازم يكون فيه التصريح عن الدالة **(Prototype)**.

## أول شيء: main.h (ملف التصريحات)

- هذا الملف يحتوي على التصريح (Prototype) للدالة **\_isupper**.
- التصريح ضروري عشان الملفات الثانية تقدر تستعمل الدالة بدون مشاكل.

```

#ifndef PWD_H
#define PWD_H

int _isupper(int c); /* سمعن الدالة */

#endif

```

## #ifndef MAIN\_H

- هذا يسمى "حارس التضمين" (Include Guard)، يمنع تضمين الملف أكثر من مرة لو تم استدعاؤه في عدة أماكن.

## #define MAIN\_H

- إذا لم يكن **MAIN\_H** معرفاً، يتم تعريفه، وهذا يضمن أن محتوى الملف يتم تحميله مرة واحدة فقط.

## int \_isupper(int c);

- هذا التصريح للدالة **\_isupper**، اللي راح نكتبها بعد شووي.
- التصريح يعني أن الدالة:

- تستقبل (int c) حرف كعد صريح.

- ترجع (0) أو (1).

## #endif

- هذه تنتهي الحارس **#ifndef** حتى لا يتم تضمين الملف أكثر من مرة.

## ثانية: isupper.c-0 (ملف تنفيذ الدالة)

- هنا نكتب الدالة **\_isupper()** التي تتحقق إذا كان الحرف كبير (Uppercase).
- ملف **isupper.c-0**

```

#include "main.h"

int _isupper(int c)
{
    // Check if character is uppercase
    if ('A' <= c <= 'Z') // ASCII code range
        return 1;
    else
        return 0;
}

```

## #include "main.h"

- هذا يستدعي ملف **main.h** اللي فيه التصريح للدالة.

## \*/ إلى /

- هذا تعليق توسيقي يوضح وظيفة الدالة، ويستخدمه المبرمجون لفهم الكود.

## int \_isupper(int c)

- تعريف الدالة **\_isupper** اللي تستقبل متغير **c** يمثل الحرف.

- الدالة ترجع 1 إذا كان الحرف كبير، و 0 إذا كان صغير أو أي شيء آخر.

## if (c &gt;= 'A' &amp;&amp; c &lt;= 'Z')

- هذا شرط يتحقق إذا كان **c** يقع بين الأحرف الكبيرة في كود A = 65 إلى Z = 90.

## return (1);

- لو الشرط تحقق (يعني **c** حرف كبير)، ترجع 1.

## else return (0);

- لو الشرط ما تتحقق (يعني **c** ليس حرف كبير)، ترجع 0.

**ثالثاً: main.c-0 (ملف الاختبار)**

```
#include "main.h"
#include <stdio.h>

int main(void)
{
    char c;
    c = 'A';
    printf("%c: %d\n", c, _isupper(c)); /* لازم ترجع 1 */
    c = 'a';
    printf("%c: %d\n", c, _isupper(c)); /* لازم ترجع 0 */
    return (0);
}
```

**#include "main.h"**  
نضمن أن main.h مستدعى علشان يقرأ التصريح للدالة.

**#include <stdio.h>**  
تضمين مكتبة stdio.h علشان نستخدم printf().

**int main(void)**  
الدالة main() اللي راح تنفذ الكود وتخبر .\_isupper()

**char c;**  
متغير c لتخزين الحروف اللي بنخبرها.

**c = 'A';**

نخزن الحرف A في c.

**printf("%c: %d\n", c, \_isupper(c));**  
طبع الحرف A والنتيجة من \_isupper(A).  
لأن حرف A كبير، المفروض ترجع 1.

هي دالة في C تستخدم لطباعة النصوص والناتج على الشاشة.

**printf:** هذا هو نمط "%c: %d\n":

**%c:** يعني أنا نريد طباعة حرف (character)، والحرف اللي بنطبعه هو c اللي هو المتغير في البرنامج.

**%d:** يعني أنا نريد طباعة رقم صحيح (integer)، وهذا الرقم هو نتيجة الدالة \_isupper(c) اللي ترجع إما 1 أو 0.

**\n:** يعني سطر جديد (new line)، اللي هو يضيف سطر جديد بعد الطباعة.

**c:** هذا المتغير اللي يحمل الحرف اللي بنعمل عليه الفحص في دالة \_isupper. في السطر الأول من البرنامج مثلاً، c هو 'A'.

**c = 'a';**

نخزن الحرف a في c.

**printf("%c: %d\n", c, \_isupper(c));**  
طبع الحرف a والنتيجة من \_isupper(a).  
لأن حرف a صغير، المفروض ترجع 0.

**return (0);**

ترجع 0 للدالة على نجاح البرنامج.

**رابعاً: تجميع الكود**

**gcc** \* هذا المترجم (compiler) اللي يحول الكود إلى ملف تنفيذى.

**-Wall -pedantic -Werror -Wextra-** \* هذه خيارات تعديدية تجعل gcc يظهر أي أخطاء برمجية.

**-std=gnu89-** \* تستخدم معيار gnu89 لتواافق الكود مع الأنظمة القديمة.

**main.c 0-isupper.c-0** \* هذه ملفات المصدر اللي راح تترجمها.

**0-isupper** \* تعنى أن اسم الملف التنفيذي الناتج هو 0-isupper.

\* تعنى أن اسم الملف التنفيذي الناتج هو 0-isupper.

## Q1:

```
1. isdigit
Write a function that checks if a digit (0 through 9)
• Prototype: int _isDigit(int c);
• Returns: 1 if c is a digit
• Returns: 0 otherwise
// C standard library provides a similar function isdigit. Has been added to show more.

#include <cs50.h>
#include "main.h"
#include <ctype.h>
#include <cs50.h>

int _isDigit()
{
    char c;
    c = '1';
    if (_isDigit('0') && _isDigit('1') && _isDigit('2') && _isDigit('3') && _isDigit('4') && _isDigit('5') && _isDigit('6') && _isDigit('7') && _isDigit('8') && _isDigit('9'))
        return (1);
    else
        return (0);
}
```

السؤال هذا يطلب منك كتابة دالة اسمها `_isDigit` تتحقق إذا كان الحرف اللي يدخل للدالة هو رقم من الأرقام من 0 إلى 9.

**المطلوب:**

- تكتب دالة تتحقق إذا كان الحرف المدخل هو رقم.
  - لو كان الحرف رقم (مثل ... 0, 1, 2... الخ)، الدالة ترجع 1.
  - لو كان الحرف مو رقم (مثلاً: حرف a أو أي حرف آخر)، الدالة ترجع 0.
- يعني ببساطة، لو دخلت الدالة رقم من 0 إلى 9، ترجع 1، وإذا دخلت حرف أو أي شيء ثاني غير الأرقام، ترجع 0.

**طريقة الحل:**

- نعرف أن الأرقام من 0 إلى 9 في جدول ASCII تبدأ من 48 (لرقم 0) وتنتهي عند 57 (لرقم 9).

- إذا كان الحرف المدخل بين هذه القيم، يعني أنه رقم.

**طريقة الكتابة:** تكتب دالة `_isDigit` تتحقق إذا كان الحرف المدخل يقع بين القيم 48 و 57.

- إذا كان صحيح، ترجع 1.
- إذا كان غير صحيح، ترجع 0.

**الخطوات:**

- نستخدم الشرط (`if`) للتحقق إذا كان الحرف بين الأرقام.

- إذا تحقق الشرط، ترجع 1 (لأنه رقم)، وإذا لم يتحقق ترجع 0 (لأنه مو رقم).

```
include "main.h"

// ...
// هذه تتحقق إذا كانت الحرف المدخل هو رقم
// + _isDigit(c);
// ...
// لو كانت الحرف رقم، 0 لو كانت غير رقم
// ...

int _isDigit(int c)
{
    if ((c >= '0' && c <= '9')) // == 9 && c <= '0')
        return (1); // نفس الحرف رقم
    else
        return (0); // نفس الحرف غير رقم
}

في الكود 18، راجع بغير الحرفين '0' و '9'
```

1. الشرط (`c >= '0' && c <= '9'`)

- إذا كان الحرف `c` يقع بين الأرقام 0 و 9، معناه أنه رقم.

• لاستخدام '0' و '9' بدلاً من القيم الرقمية، لأنهما تمثلان القيم في جدول ASCII

`:;return (1);`

- إذا تحقق الشرط، ترجع 1 يعني الحرف هو رقم.

`:;return (0);`

- إذا لم يتحقق الشرط، ترجع 0 يعني الحرف ليس رقم.

**النتيجة:**

• إذا كان الحرف هو '0' أو أي رقم آخر بين 0 و 9 الدالة ترجع 1

• إذا كان الحرف غير ذلك (مثل 'a') الدالة ترجع 0

**النتيجة المنشورة على الشاشة هي:**

```
0-1
a-0
```

**ملاحظة :** سوينا ملف

وحطينا فيه نفس الكود اللي في الصوره اللي قبل مع تغيير اسم المتغير `main.h`

```
include "main.h"
...
// _isDigit - Checks if a character is a digit (0-9).
// Do the character is blank.
// Returns 1 if it is a digit, 0 otherwise.
int _isDigit(int c)
{
    if (c >= '0' && c <= '9')
        return (1);
    else
        return (0);
}
```

**النتيجة:**

- إذا كان الحرف هو '0' أو أي رقم آخر بين 0 و 9 الدالة ترجع 1
- إذا كان الحرف غير ذلك (مثل 'a') الدالة ترجع 0

## Q2

```
2. Collaboration is multiplication
Write a function that multiplies two integers.
• Prototype: int mul(int a, int b);
jallen@allen:~/S/int_0/main$ gcc -Wall -pedantic -Werror -Wextra -Wextra -Wsign-compare -fmain.c -o 2-mul
jallen@allen:~/S/int_0/main$ ./2-mul
30032
jallen@allen:~/S/int_0/main$
```

Repo

- GitHub repository: [https://github.com/leensaleh/cse-level-programming](#)
- Directory: [www\\_function\\_main\\_level](#)
- File: [2-mul.c](#)

السؤال يطلب منك تبرمجة دالة **function** بلغة C تقوم بضرب عددين صحيحين (integers) وترجع ناتج الضرب.

وش المطلوب بالضبط؟

1- تكتب دالة اسمها **mul**، وتكون بالشكل التالي:

**int mul(int a, int b);**

تأخذ عددين صحيحين (a) و (b).

ترجع نتيجة ضربهم (a \* b).

2- تحفظ الكود في ملف اسمه **2-mul.c**.

الكود الدالة

**2-mul.c**

**#include "main.h"**

**\*\*/**

```
#include "main.h"
/**
* mul - Multiplies two integers.
* @a: The first integer.
* @b: The second integer.
*
* Return: The product of a and b.
*/
int mul(int a, int b)
{
    return (a * b);
}
```

\*mul - Multiplies two integers.

@ \*a: The first integer.

@ \*b: The second integer.

\*

\*Return: The product of a and b.

/\*

int mul(int a, int b)

}

return (a \* b);

{

الكود 2-main.c

#include "main.h"

هذا السطر يستدعي ملف الرأس **header** ملف الرأس **main.h** ، والذي من المفترض أنه يحتوي على تعريفات الدوال المستخدمة في البرنامج، مثل **mul**.

السطر الثاني:

**#include <stdio.h>**

يستخدم مكتبة **stdio.h** ، وهي المكتبة المسئولة عن وظائف الإدخال والإخراج مثل **printf**.

(Documentation Comment)

**/\*\***

\* main - check the code

\*

\* Return: Always 0.

**\*/**

```
#include "main.h"
#include <stdio.h>

/**
* main - check the code
*
* Return: Always 0.
*/
int main(void)
{
    printf("%d\n", mul(98, 1024));
    printf("%d\n", mul(-402, 4096));
    return (0);
}
```

- ◆ هذا مجرد تعليق توضيحي يصف وظيفة الدالة **main**.
- ◆ يقول إن هذه الدالة تستخدم لاختبار الكود.
- ◆ أيضاً، يوضح أن الدالة سترجع دانما **0**.

#### تعريف الدالة:

**int main(void)**

- ◆ هذا هو المكان الذي يبدأ فيه تنفيذ البرنامج.
- ◆ الدالة **main** لا تأخذ أي **arguments** (مدخلات).
- ◆ ارجاعها يكون دانماً عدداً صحيحاً (**int**).

#### السطر الأول داخل الدالة:

**printf("%d\n", mul(98, 1024));**

- ◆ هي دالة لطباعة النصوص على الشاشة.
- ◆ تستدعي الدالة **mul**، والتي تقوم بضرب الرقمين  $1024 \times 98$ . ثم تطبع النتيجة بصيغة **%d** (عدد صحيح).
- ◆ هو رمز سطر جديد (يعني تطبع النتيجة ثم تنزل لسطر جديد).

#### السطر الثاني داخل الدالة:

**printf("%d\n", mul(-402, 4096));**

- ◆ تستدعي الدالة **mul** مرة أخرى، لكن هذه المرة بارقام مختلفة. (-402 × 4096).
- ◆ تطبع النتيجة ثم تنزل لسطر جديد.

**return (0);**

- ◆ يعيد **0** إلى نظام التشغيل، مما يدل على أن البرنامج انتهى بنجاح.
- ◆ هذا شيء شائع في برامج **C**، حيث **0** يعني "نجاح"، وأي قيمة أخرى قد تشير إلى وجود خطأ.

## Q3

```

3.The numbers speak for themselves
=====
Write a function that prints the numbers, from 0 to 9, followed by a new line.
* Prototype: void print_numbers(void);
* You can only use _putchar_ twice in your code
=====
#include <stdio.h>
#include "main.h"

void
print_numbers(void)
{
    /* print - check the code
     * 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
     */
    int i;
    for (i = 0; i < 10; i++)
        putchar(numbers[i]);
    /* print_newline();
     */
}

int main()
{
    /* print - check the code
     * 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
     */
    print_numbers();
    /* print_newline();
     */
}

Reps:
  - GitHub repository: https://github.com/leensaleh/Computer_Programming
  - Created: 2020-08-20, Last updated: 2020-08-20
  - File: 3-print_numbers.c

```

السؤال يطلب منك تبرمجة دالة بلغة C تطبع الأرقام من **0** إلى **9**، وبعدها تنزل سطر جديد.

وشن المطلوب بالضبط؟

تكتب دالة اسمها **print\_numbers**، وتكون بالشكل التالي:

**void print\_numbers(void);**

ما تستقبل أي مدخلات.

طبع الأرقام من **0** إلى **9**.

بعد الأرقام، تنزل سطر جديد.

1. تستخدم **putchar** مرتين فقط في الكود (ما يمنع تستخدم **printf** أو **putchar** أكثر من مرتين).

**header main.h.** غالباً يكون معرف في ملف **\_putchar**

**\_putchar** يستخدم لطباعة حرف واحد فقط في كل مرة.

2. تحفظ الكود في ملف اسمه **3-print\_numbers.c**.

```

#include "main.h"
=====
void print_numbers() {
    /* print_numbers - Prints the numbers from 0 to 9, followed by a new line
     */
    void print_numbers(void)
    {
        char numbers[] = "0123456789";
        int i;
        for (i = 0; numbers[i] != '\0'; i++)
            putchar(numbers[i]);
    }
}

```

#### كود : 3-print\_numbers.c

```

#include "main.h"
=====
void print_numbers()
{
    /* print_numbers - Prints the numbers from 0 to 9, followed by a new line
     */
    void print_numbers(void)
    {
        char numbers[] = "0123456789";
        int i;
        for (i = 0; numbers[i] != '\0'; i++)
            putchar(numbers[i]);
    }
}

```

\* لغات اضافتها: 2020-08-20، ولكن قد لا يتوفر في كل لغة. إذا لم يتوفر في لغة ما، فلن يظهر في الكود.

لذا استخدمنا مجموعة المتصور (fubar) حيث يحتوي على الأرقام 0-9، ونحوه، ويعطيها ملائمة ملائمة واحدة.

char numbers[] = "0123456789";

```
for (int i = 0; i < 10; i++) {
    putchar(numbers[i]);
}
```

- \* هنا عرفنا مصفوفة (array) اسمها `numbers`, وهي تحتوي على الأرقام من 0 إلى 9، وبعدين سطر جديد.
- \* المصفوفة هذى هي عبارة عن سلسلة تنصيحة (String) متسلسلة، أول عنصر فيها هو "0"، وثاني عنصر هو "1"، وهكذا إلى "9". وبعدين عندنا سطر جديد في النهاية ("").
- \* هنا تعرف الدالة `putchar` من نوع `int` (عدد صحيح).
- \* هذا المتغير ي المستخدمة عشان نمر على المصفوفة `numbers`

كود اللي مكتوب بالسؤال :

السطر الأول:

```
#include "main.h"
```

- \* هذا السطر يستدعي ملف **main.h** اسمه **header**.
- \* الملف هذا يحتوي على تعاريفات دوال ومتغيرات تحتاجها في البرنامج، زي الدالة **print\_numbers** اللي بنسخدمها في هذا البرنامج.

التعليق التوضيحي:

```
/**
```

```
* main - check the code
```

```
*
```

```
* Return: Always 0.
```

```
*/
```

- \* هذا تعليق يوضح وظيفة الدالة.

السطر الثاني:

```
int main(void)
```

- \* هذا هو تعريف الدالة.

السطر الثالث:

```
{
```

```
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_numbers();
    return (0);
}
```

- \* يقول ان الدالة هذه جايه عشان تتحقق من الكود (ختبره).

- \* يعني الدالة هذى ترجع القيمة **0** دائمًا، وال**0** هنا تعنى أن البرنامج انتهى بنجاح.

السطر الثاني:

- \* يعني أن الدالة **main** ترجع قيمة من نوع **int** عدد صحيح.

- \* يعني أن الدالة ما تستقبل أي مدخلات (الـ **void**) يعني "ما في مدخلات".

السطر الثالث:

- ◆ هذا بداية جسم الدالة **main**، يعني كل الكود اللي بيشغل داخل الدالة ينكتبه هنا.

السطر الرابع:

**print\_numbers();**

- ◆ هنا نداء إلى الدالة **print\_numbers**. الدالة هذه اللي عرفناها في الملف الثاني **3-print\_numbers.c**، وهي المسؤولة عن طباعة الأرقام من 0 إلى 9.

- لما **print\_numbers** تنادي، يتم تنفيذ الكود اللي فيها (طباعة الأرقام).

السطر الخامس:

**return (0);**

- ◆ بعد ما ننتهي من تنفيذ الكود في **main**، نرجع قيمة **0** إلى نظام التشغيل.

- يعني أن البرنامج انتهى بشكل طبيعي وبدون مشاكل.

- في C ، إرجاع **0** في **main** يعتبر إشارة ناجح.

السطر السادس:

}

- ◆ نهاية جسم الدالة **main**. هنا تكون انتهينا من تعریف الدالة **main**.

كود ال **main.h** اضفنا عليه تعريف ثانٍ صار كذا :

```
#ifndef MAIN_H
#define MAIN_H
int _putchar(char c);

void print_numbers(void);/* Function prototype*/
#endif /* MAIN_H*/
```

#### Q4:

4.1 Define print\_numbers and sign.

Write a function that prints the numbers from 0 to 9, followed by a newline.

- Prototype: `void print_most_numbers(void)`
- Do not print 2 and 4.
- You can only use `_putchar` twice in your code.

```
/** 
 * main - check the code
 * 
 * Return: always 0.
 */
int main(void)
{
    print_most_numbers();
    return (0);
}

int_putchar(int character)
{
    write(1, &character, 1);
}
```

Repo:
 

- GitHub repository: [fallah-leen-fallah-leen-programming](#)
- DIRECTORY: [work/functions/most\\_numbers](#)
- FILE: [most\\_numbers.c](#)

لسؤال هذا بطلب منك تبرمجة دالة بلغة C تطبع الأرقام من 0 إلى 9، ولكن مع استثناء الرقمين 2 و 4 . وفي نفس الوقت، لازم تستخدم **\_putchar** مرتين فقط.

وش المطلوب بالضبط؟

1. كتابة دالة باسم **print\_most\_numbers**، يكون شكلها كذا:

**void print\_most\_numbers(void);**

لازم تطبع الأرقام من 0 إلى 9، مع استثناء الرقمين 2 و 4، يعني ما تطبعهم.

بعد الأرقام، لازم تنزل سطر جديد **\n**.

2. استخدام **\_putchar** مرتين فقط:

**\_putchar** تستخدمها لطباعة حرف واحد في كل مرة، فلازم تنظم الكود عشان تستخدمها مرتين فقط.

3. تكتب الكود في ملف اسمه **4-print\_most\_numbers.c**.

// طبعاً سوينا سكريبت المين والمين اللي موجوده بالسؤال

```

print_most_numbers.c
#include "main.h"
void print_most_numbers(void)
{
    int i;
    for (i = 0; i < 10; i++)
    {
        if (i != 2 && i != 4)
            putchar(i + '0');
    }
    putchar('\n');
}

```

كود ملف 4-print\_most\_numbers.c

```

#include "main.h"
/*
 *print_most_numbers - Prints numbers from 0 to 9, except 2 and 4.
 */
void print_most_numbers(void)
{
    int i;
    for (i = 0; i < 10; i++)
    {
        if (i != 2 && i != 4)
            putchar(i + '0');
    }
    putchar('\n');
}

```

#include "main.h"

- هذا السطر يستدعي ملف main.h، اللي غالباً يكون فيه تعریفات الدوال المستخدمة، مثل ()\_putchar() اللي نستخدمه هنا عشان نطبع الحروف.

void print\_most\_numbers(void)

- هذا دالة اسمها print\_most\_numbers، وهي دالة ما تستقبل أي بارامترات (void) وما ترجع قيمة.

int i;

- هنا عَرَفْنا متغير من نوع int، اللي راح نستخدمه كعداد في الحلقة التكرارية.(for)

for (i = 0; i < 10; i++)

- هذا حلقة for، وظيفتها تمر على الأرقام من 0 إلى 9.
- تفاصيلها:

0 = أَنْدَأْ من الرقم 0 ↩ •

10 > يَسْتَمِرُ الحلقة كل ما كان أَقْلَ من 10 ↩ •

++ زيد كل تكرار، نزيد زيمقدار 1 ↩ •

بالنالي، زراح يمر بالقيم:  
0, 1, 2, 3, 4, 5, 6, 7, 8, 9

if (i != 2 && i != 4)

- هنا نتحقق إذا كانت إما 2 و إما 4، يعني أي رقم عدا 2 و 4 نطبع.

ليش استخدمنا ؟&& (AND)

- لأنه لازم يكون الرقم إما 2 و إما 4 بنفس الوقت عشان يطبع.

\_putchar(i + '0');

- وظيفته بطبع الحرف.

i + '0' معناها تحول الرقم إلى حرف نصي ( لأنه في ASCII ) ، 0 هو الحرف 48، و 1 هو 49 وهكذا.

فإذا كان 3 = i، راح يطبع '3'، وإذا كان 5 = i، راح يطبع '5'.

`_putchar('\n');`

◆ بعد ما نخلص طباعة الأرقام، نحط سطر جديد (\n) عشان تنزل الكتابة للسطر اللي بعده.

## Q6:

```
6. The straight distance between two points is a straight line.
What does the following code do?
• Print the character '_'.
• Print the character '_' n times.
• When n < 0, the character '_' should be printed.
• The function ends with a '\n'.
• If n = 0 or less, the function immediately prints '_'.
#include <cs50.h>
#include <stdio.h>
int main()
{
    // Prompt user for number of times character '_' should be printed
    int n = getint("How many times? ");
    // Print character '_'
    for (int i = 0; i < n; i++)
    {
        putchar('_');
    }
    // Print new line
    putchar('\n');
}
```

السؤال بيغى منك تبرمجة دالة ترسم خط مستقيم باستخدام الحرف `_` في التيرمينال. خليني أوضح لك الفكرة ببساطة:

وش المطلوب بالضبط؟

تسوي دالة اسمها `print_line` تأخذ عدد صحيح `n` كمدخل.

إذا كان `n` أكبر من 0، تطبع `_` بهذا العدد `n` في سطر واحد.

إذا كان `n` أقل من أو يساوي 0، تطبع بس سطر جديد \n بدون أي شيء ثاني.

لازم تستخدم الدالة `putchar` للطباعة وما تستخدم أي دالة ثانية.

```
#include "main.h"
/*
 * print_line - Draws a straight line in the terminal.
 * @n: Number of times the character '_' should be printed.
 */
void print_line(int n)
{
    int i;
    if (n > 0)
    {
        for (i = 0; i < n; i++)
        {
            putchar('_');
        }
        putchar('\n');
    }
}
```

كود ملف 6-print\_line.c

```
#include "main.h"
/*
 * print_line - Draws a straight line in the terminal.
 * @n: Number of times the character '_' should be printed.
 */
void print_line(int n)
{
    int i;
    if (n > 0)
    {
        for (i = 0; i < n; i++)
        {
            putchar('_');
        }
        putchar('\n');
    }
}
```

// كود اللي حاطينه بالسؤال

```
#include "main.h"
/*
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_line(0);
    print_line(2);
    print_line(10);
    print_line(-4);
    return (0);
}
```

#include "main.h"
/\*
 \* main - check the code
 \*
 \* Return: Always 0.
 \*/
int main(void)

{

print\_line(0);

```

    print_line(2);
    print_line(10);
    print_line(-4);
    return (0);
}

```

// ملاحظه عدلنا على اسم زى العادة main.h

### Q7 :

السؤال يطلب منك كتابة دالة (print\_diagonal) تطبع خط قطرى (\) في التيرمنال.

التفاصيل اللي لازم تنتبه لها:

1. الدالة لازم تستخدم **فقط putchar** للطباعة.

2. إذا كان n = 5، نطبع \ بشكل مائل كذا:

\  
\  
\  
\  
\

3. لازم ينتهي الطباعة بسطر جديد \n.

4. إذا 0 <= n < 5، نطبع سطر جديد فقط بدون أي .

```

#include "main.h" // شرح كود اللي موجود بالسؤال

/*
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_diagonal(0);
    print_diagonal(2);
    print_diagonal(10);
    print_diagonal(-4);
    return (0);
}

#include "main.h" // شرح كود اللي موجود بالسؤال

/*
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_diagonal(0);
    print_diagonal(2);
    print_diagonal(10);
    print_diagonal(-4);
    return (0);
}

```

: 7-print\_diagonal.c شرح كود الملف

```
#include "main.h"
*/
* print_diagonal - Draws a diagonal line on the terminal
* @n: The number of times the character \ should be printed.
*/
void print_diagonal(int n)
{
int i, j;
if (n <= 0)
{
_putchar('\n');
return;
}
for (i = 0; i < n; i++)
{
for (j = 0; j < i; j++)
{
_putchar(' ');
}
_putchar('\\');
_putchar('/');
}
}
```

```
#include "main.h"
/*
* print_diagonal - Draws a diagonal line on the terminal
* @n: The number of times the character \ should be printed.
*/
void print_diagonal(int n)
{
int i, j;
if (n <= 0)
{
_putchar('\n');
return;
}
for (i = 0; i < n; i++)
{
for (j = 0; j < i; j++)
{
_putchar(' ');
}
_putchar('\\');
_putchar('/');
}
}
```

• السطر الأول:

```
#include "main.h";
```

هذا السطر يستعين بال ملف main.h ، الذي غالباً يحتوي على تعریف الدالة \_putchar() . لأننا نستخدمها هنا للطباعة

• تعریف الدالة:

```
void print_diagonal(int n);
```

هذا عریفنا دالة print\_diagonal() التي تأخذ عدد صحيح (n) ، وتمثل عدد مرات طباعة \ بشكل مائل.

• تعریف المتغيرات:

```
int i, j;
```

عندنا متغيرين (i) و (j) . نستخدمهم في الحلقات (for) . يمكن التحكم في عدد الأسطر والمسافات .

• التحقيق إذا (i) أقل من أو يساوي (n) :

```
if (n >= 0)
{
_putchar('\n');
return;
}
```

• شرح بحثكوا:

- كل ما زلنا سطر (i) ، يعني نضيف عدد معين من المسافات (space) قبل ما نطبع خط بطول صفر أو أقل . وهي حالة خاصة .
- الحل ، نطبع سطر جديد (n) فقط ونطلع من الدالة (return)

• الحلقة الداخلية (for) التي تضيف المسافات:

```
for (j = 0; j < i; j++)
{
_putchar(' ');
}
```

• وظيفة (for) :

- كل ما زلنا سطر (i) ، يعني نضيف عدد معين من المسافات (space) قبل ما نطبع خط بطول صفر أو أقل . وهي حالة خاصة .
- المسافة (space) وحدة (unit) من المسافات (spaces) .
- وهذا ...
- لهذا من الأمثل ، يعني بذلك (for) مرة ، كل مرة أدخل سطر جديد .
- في كل سطر ، لازم تضيف مسافات قبل ما تطبع (i) عشان تكون مائل .

```
_putchar('\'')
```

- \* طباعة \ ونهاية المسطر:

- \* بعده ما تطبع المسافات، تطبع \ عشان يكون بداية الخط قطرى.
- \* بعد \ تطبع سطر جديد (\\) عشان بعد المسطر اللي بعده

مثال على التنفيذ:  
لو أدخلنا المدخلات  
1 2 3 4 5  
نكون نطبع  
1  
2  
3  
4  
5

نلاحظ إن \ يكون تحت سطر جديد، وهذا كل مرة يزيد فحمة \، تزيد المسافات التي قبل \.

## الزيادة

- الدالة تطبع خط قطرى (\) مائل باستخدام \_putchar\_.
- نستخدم حلقتين for ، وحدة لمسافات والثانية لرسم الخط
- إذا 0 <= n ، تطبع بس سطر جديد (\).

```
for (i = 0; i < n; ++i)
```

هو المتغير اللي يمثل عدد الأسطر اللي راج لرسمها.  
يبدأ من 0، ويزيد كل مرة (++) إلى أن يصل إلى n، وهذا يعني إنه يتحكم بعدد عزات الطباعة.

- \* مثال لو 5

المسطر المطبوع	قيمة i
\	1 + 0
\	1 + 1
\	1 + 2
\	1 + 3
\	1 + 4

كل مرة \ يزيد، تضيف مسافة إضافية قبل \، عشان يكون الخط مائل.

## المتغير i (يتحكم بعد المسافات)

```
for (i = 0; i < 5; ++i)
{
    putchar('\\');
}
```

- هو المتغير اللي يتحكم بعد المسافات قبل طباعة \.
- الحلقة تبدا من 0 إلى 5، وهذا يعني إنه راج يطبع مسافة (+) بعدد يساوي 5.

المسطر المطبوع	كم مرة تتشغل i (عدد المسافات)	قيمة i
\	0 عزات	1 + 0
\	1 عزات	1 + 1
\	2 عزات	1 + 2
\	3 عزات	1 + 3
\	4 عزات	1 + 4

كلما زادت i، زادت عدد المسافات اللي يضيفها \ قبل طباعة \.

## الزيادة

- يتحكم في عدد الأسطر اللي لرسمها.
- يتحكم بعدد المسافات اللي يضيفها \ قبل كل سطر.
- مع كل زيادة في i، يزيد أكثر عشان يخل الخط مائل.

يعني آخر i يحدد كم مرة ينطبع \، و i يحدد كم مسافة قبل كل \.

## Q8:

```
#include <cs50.h>
#include "main.h"

void print_square(int size)
{
    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            _putchar('*');
        }
        _putchar('\n');
    }
}

int main(void)
{
    print_square(3);
    return 0;
}
```

السؤال يعني منك تكتب دالة تطبع مربع باستخدام الحرف #. عشان نفهم السؤال بشكل أفضل، خليني أشرح لك المطلوب بطريقة مبسطة وباللهجة السعودية:

**المطلوب في السؤال:**

1. كتابة دالة اسمها **print\_square** تستقبل متغير واحد **size** هو حجم المربع (مثلاً 2 أو 3 أو 4 وهذا).
2. المربع لازم يكون مكون من حروف #، وكل سطر في المربع لازم يحتوي على #.
3. بعد ما يكتمل المربع، لازم تطبع سطر جديد (\n).
4. إذا كانت قيمة **size** صفر أو أقل، لازم تطبع فقط سطر جديد بدون أي مربع.

الشرح بالتفصيل الكود اللي قدمك:

**1. #include "main.h"**

- هنا الكود يضمن وجود الملف "main.h" في الكود. هذا الملف يحتوي على تعريفات الدوال اللي بنسخدمها في الكود مثل دالة **\_putchar** اللي مسؤولة عن طباعة الأحرف.

**2. التعليق في بداية الكود**

```
/*
 * print_square - prints a square, followed by a new line
 * @size: the size of the square
 */
```

- التعليق هذا يوضح وظيفتك في هذه الدالة :

**print\_square:** ○ اسم الدالة اللي بتقوم بطباعة مربع.

○ متغير **size** هو حجم المربع، يعني عدد الأحرف اللي بيتم طباعتها في كل صف (والعدد نفسه بيحدد عدد الصفوف).

**3. الدالة نفسها:**

**void print\_square(int size)**

- هنا الدالة **print\_square** هي دالة تأخذ متغير واحد وهو **size** اللي هو حجم المربع.

**الشرط الأول:**

```
if (size <= 0)
{
    _putchar('\n');
    return;
}
```

- هذا الشرط يتحقق أولًا إذا كانت القيمة المدخلة لـ **size** أقل من أو تساوي صفر. يعني لو حاولت تطبع مربع بحجم صفر أو حجم سالب، ما راح يكون في مربع يظهر، فقط هيطبع سطر جديد باستخدام **\_putchar("\n")**.

**الـ for الأولى:**

```
for (int i = 0; i < size; i++) // Loop through each row
```

- هنا عندنا **for loop** ينكرر **size** مرات. في كل مرة، نقوم بطباعة صف واحد من المربع. الـ **i** في هذا الـ **for loop** يمثل عدد الصفوف.

**الـ for الثانية:**

```
for (int j = 0; j < size; j++) // Loop through each column
```

- دايرك الدايرك الأولى، في عندنا **for loop** ثانية تكرر نفسها **size** مرات . هذى مسؤولة عن طباعة الدايرك كل عمود. الدايرك هنا تمثل الأعمدة دايرك كل صاف.

طباعة الدايرك:

```
putchar('#');
```

- كل مرة تكرر فيها الحلقة الداخلية (الد **for** الثانية)، يتم طباعة الدايرك # باستخدام الدالة **putchar**. هذه هي الوحدة الأساسية لبناء المربع.

طباعة سطر جديد بعد كل صاف:

```
putchar('\n');
```

- بعد ما تخلص الد **for loop** الثانية ( اللي طبعت جميع الأعمدة في صاف واحد)، لازم نطبع سطر جديد \n\ باستخدام **putchar** عشان نبدأ صاف جديد من المربع.

النتيجة النهائية:

إذا كان size > 0، راح يطبع مربع مكون من size صفوف وكل صاف يحتوي على size من الدايرك #.

إذا كانت القيمة المدخلة لـ size صفر أو أقل، راح يطبع سطر واحد فقط.

```
#include "main.h"
```

```
*/
* print_square - prints a square, followed by a new line
@ * size: the size of the square
/*
void print_square(int size)
{
    if (size <= 0)
    {
        putchar('\n');
        return;
    }

    for (int i = 0; i < size; i++) // Loop through each row
    {
        for (int j = 0; j < size; j++) // Loop through each column
        {
            putchar('#');
        }
        putchar('\n'); // Print a new line after each row
    }
}
```

## 9. Fizz-Buzz

The "Fizz-Buzz test" is an interview question designed to help filter out the 99.5% of programming job candidates who can't seem to program their way out of a wet paper bag.

Write a program that prints the numbers from 1 to 100, followed by a new line. But for multiples of three print `Fizz` instead of the number and for the multiples of five print `Buzz`. For numbers which are multiples of both print `FizzBuzz`.

- Each number or word should be separated by a space
- You are allowed to use the standard library

```
salim@elbert:~/git/wall-pediatric-server-wsenv -rw-rw-rw- 1 Fizz_Buzz.c -> 0-Fizz_Buzz
salim@elbert:~/git/Fizz_Buzz
1-2 Fizz 3 Buzz Fizz 5 Fizz 6-10 BuzzBuzz 11-12 FizzBuzz 13-17 Fizz 18 Buzz Fizz 20-24 Fizz Buzz 26 Fizz 28-29 Fizz
Buzz 31-32 Fizz 34 Buzz Fizz 37-38 Fizz Buzz 41 Fizz 43-44 FizzBuzz 48-49 FizzBuzz 52-53 FizzBuzz 56-59 FizzBuzz
60-63 FizzBuzz 68-69 FizzBuzz 71-72 FizzBuzz 73-74 FizzBuzz 76-77 Fizz 79 Buzz Fizz 82-83 FizzBuzz
1-98 FizzBuzz 91-93 Fizz 94 Buzz 95-97 Fizz 98 FizzBuzz
salim@elbert:~/git/Fizz_Buzz
```

### Steps:

- GitHub repository: [https://github.com/leensalaleh/low-level\\_programming](https://github.com/leensalaleh/low-level_programming)
- Directory: `more_fundamentals/exercices/lego`
- File: `9-Fizz_Buzz.c`

## Q9:

السؤال يبي منك تكتب برنامج يقوم بطباعة الأرقام من 1 إلى 100، لكن مع استبدال بعض الأرقام بكلمات معينة حسب القواسم:

1. إذا الرقم كان قابل للقسمة على 3، بطبع "Fizz" بدل الرقم.
2. إذا الرقم كان قابل للقسمة على 5، بطبع "Buzz" بدل الرقم.
3. إذا الرقم كان قابل للقسمة على 3 و 5 مع بعض، بطبع "FizzBuzz" بدل الرقم.

**الشروط واضحة:**

- الأرقام أو الكلمات لازم تكون مفصولة بمسافة.

لو كان الرقم 1 أو 2 أو أي رقم ما ينطبق عليه أي من الشروط السابقة، بطبع الرقم نفسه.

**المطلوب منك:**

- كتابة البرنامج بحيث يعرض الأرقام من 1 إلى 100، مع تطبيق الشروط المذكورة.
- إذا الرقم قابل للقسمة على 3، بطبع "Fizz".
- إذا الرقم قابل للقسمة على 5، بطبع "Buzz".
- إذا الرقم قابل للقسمة على 3 و 5 مع بعض، بطبع "FizzBuzz".
- لازم كل شيء يكون مفصول بمسافة.

**كود الملف :**

```
<include <stdio.h#
```

```
**/
```

main - prints the numbers from 1 to 100, replacing multiples of 3 and 5 \*

```
int main(void)
{
    int i;
    for (i = 1; i <= 100; i++)
    {
        if (i % 3 == 0 && i % 5 == 0)
            printf("FizzBuzz");
        else if (i % 3 == 0)
            printf("Fizz");
        else if (i % 5 == 0)
            printf("Buzz");
        else
            printf("%d", i);
        printf("\n");
    }
    printf("\n");
    return 0;
}
```

```
 * .Return: Always 0 *
 */
int main(void)
{
    ;int i
    for (i = 1; i <= 100; i++)
    {
        if (i % 3 == 0 && i % 5 == 0)
            ;printf("FizzBuzz");
        else if (i % 3 == 0)
            ;printf("Fizz");
        else if (i % 5 == 0)
            ;printf("Buzz");
        else
            ;printf("%d", i);
        if (i != 100)
```

```
;(" ")printf
```

```
{
```

```
:printf("\n")
```

```
;(0) return
```

```
{
```

---

```
Q10 :
```

```
"include "main.h#
```

```
**/
```

```
print_triangle - prints a triangle, followed by a new line *
```

```
size: the size of the triangle@ *
```

```
/*
```

```
void print_triangle(int size)
```

```
}
```

```
if (size <= 0)
```

```
}
```

```
:putchar('\n')
```

```
;return
```

```
{
```

```
for (int i = 1; i <= size; i++) // Loop through each row
```

```
}
```

```
Print spaces for alignment //
```

```
for (int j = 0; j < size - i; j++)
```

```
}
```

```
:(' ')putchar_
```

```
{
```

```
Print # characters for the triangle //
```

```
for (int j = 0; j < i; j++)
```

```
{
```

```
:('#')putchar_
```

```
{
```

```
Print a new line after each row //
```

```
:putchar('\n')
```

```
{
```

```
{
```

