0. Float like a butterfly, sting like a bee

Write a function that creates an array of chars, and initializes it with a specific char.

- Prototype: `char *create_array(unsigned int size, char c);`
- Returns NULL if size = 0
- Returns a pointer to the array, or NULL if it fails

```
julien@ubuntu:~/0x0a. malloc, free$ cat 0-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

/**
 * simple_print_buffer - prints buffer in hexa
 * @buffer: the address of memory to print
 * @size: the size of the memory to print
 *
 * Return: Nothing.
 */
void simple_print_buffer(char *buffer, unsigned int size)
{
    unsigned int i;

    i = 0;
    while (i < size)
    {
        if (i % 10)
        {
            putchar(' ');
```

```
#include <stdlib.h>

#include "main.h"

/**

* create_array - Creates an array of chars and initializes it

* with a specific char.

* @size: The size of the array.

* @c: The character to initialize the array with.

*

* Return: Pointer to the array, or NULL if it fails or size is 0.

*/

char *create_array(unsigned int size, char c)

{

char *arr;

unsigned int i;

if (size == 0)

return (NULL);

arr = malloc(size * sizeof(char));

if (arr == NULL)

return (NULL);

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

arr[i] = c;

return (arr);

}
```

شرح الكود :

فكرة السؤال :

السؤال وش يبي بالضبط؟

السؤال يطلب منك تكتب دالة (function) في لغة **C** تسوي التالي:

1. تنشئ مصفوفة (array) من الأحرف.(chars)
2. تملأ المصفوفة بحرف معيّن يحدده المستخدم.
3. ترجع **مؤشر (pointer) للمصفوفة**.
4. إذا كان الحجم اللي طلبه المستخدم **0**، ترجع .**NULL**
5. إذا صار فيه **مشكلة في تخصيص الذاكرة(memory allocation fails)** ، برضه ترجع .**NULL**

كيف نبرمج الدالة؟

1. إذا كان size صفر **نرجع NULL** لأن المصفوفة بدون حجم مالها داعي.
2. نحجز ذاكرة باستخدام malloc لحجم size من نوع .char
3. إذا فشل malloc يعني ما قدر يحجز ذاكرة **نرجع .NULL**
4. إذا نجح التخصيص، نملأ كل عنصر في المصفوفة بالحرف .c
5. في النهاية، نرجّع مؤشر المصفوفة.

إنشاء دالة باسم create_array، والتي تستقبل:

عدد صحيح size يمثل حجم المصفوفة.

حرف c سيتم تخزينه في جميع عناصر المصفوفة.

وظيفة الدالة:

تخصيص (Allocate) ذاكرة ديناميكية لمصفوفة من الأحرف بحجم sizeباستخدام .malloc

إيش يعني تخصيص ذاكرة ديناميكية؟(Dynamic Memory Allocation)

تخصيص الذاكرة الديناميكية يعني إنك **تحجز مساحة في الذاكرة (RAM) أثناء تشغيل البرنامج، مو قبلها.**

```
char array[10] ;
```

هنا، حجم المصفوفة ثابت ومحدد وقت كتابة الكود(Compile Time) ، وما تقدر تغيّره أثناء تشغيل البرنامج.

لكن في بعض الأحيان، ما تعرف حجم البيانات مسبقًا، فتحتاج **تحجز ذاكرة أثناء تشغيل البرنامج** باستخدام .malloc

كيف نحجز ذاكرة ديناميكية

نستخدم malloc() لحجز ذاكرة بالحجم اللي نحتاجه

ليش لازم نستخدم free()؟

بما إننا استخدمنا malloc، فالذاكرة **ما تنحذف تلقائيًا** بعد انتهاء الدالة.
لذلك، لازم نحذفها باستخدام free()حتى ما يصير **تسريب في الذاكرة.(Memory Leak)**

شرح الكود سطر بسطر :

المكتبات اللي نحتاجها :

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

#include <stdlib.h>
هذه المكتبة فيها الدالة malloc اللي نستخدمها لتخصيص ذاكرة ديناميكية (حجز ذاكرة أثناء تشغيل البرنامج).

#include "main.h"
هذا ملف الهيدر (header file) اللي غالبًا فيه تعريف (prototype) للدالة create_array عشان يقدر الملف الرئيسي (main.c) يستخدمها.

تعريف الداله :

```
char *create_array(unsigned int size, char c)
```

char *create_array الدالة ترجع **مؤشر (Pointer)** لنوع char، يعني راح ترجع عنوان (Address)لأول عنصر في المصفوفة.

unsigned int size متغير يمثل حجم المصفوفة اللي نبي ننشئها.

char c الحرف اللي بنحطّه داخل كل عناصر المصفوفة.

تعريف المتغيرات:

```
char *arr;
unsigned int i;
```

char *arr; هذا مؤشر (Pointer) من نوع char، راح نستخدمه لتخزين العنوان اللي يرجع من malloc (المصفوفة المحجوزة في الذاكرة).

unsigned int i; متغير عدّاد (counter) نستخدمه داخل for عشان نعبّي المصفوفة بالحرف c.

التحقق من size:

```
if (size == 0)
    return (NULL);
```

إذا كان size == 0، معناته المستخدم طلب مصفوفة بدون حجم، وهذا **ما له معنى**، فنرجع NULL على طول عشان نقول له **"ما سويت لك مصفوفة ."**

NULL تعني **"ما فيه عنوان ذاكرة صالح "**أو **"العملية فشلت."**

حجز الذاكرة باستخدام malloc:

```
arr = malloc(size * sizeof(char));
```

هنا قاعدين نحجز ذاكرة بحجم size مضروب في sizeof(char).

- sizeof(char) دائمًا يساوي **1بايت**، فبالتالي malloc(size * 1) تحجز size بايت من الذاكرة.
- malloc ترجع **عنوان أول مكان في الذاكرة المحجوزة**، ونخزنه داخل arr.
- إذا malloc فشلت (ما فيه ذاكرة كافية)، ترجع NULL.

التحقق من malloc:

```
if (arr == NULL)
    return (NULL);
```

إذا malloc رجّعت NULL، معناها ما قدرنا نحجز ذاكرة (الرام فل أو فيه مشكلة)، فـ **نرجع NULL** ونوقف التنفيذ.

تعبئة المصفوفة بالحرف c

```
for (i = 0; i < size; i++)
    arr[i] = c;
```

هنا عندنا for تكرارية:

- i = 0 نبدأ من أول عنصر في المصفوفة.
- i < size نكرّر حتى نوصل لآخر عنصر.
- i++ كل مرة نزود i عشان نروح للعنصر اللي بعده.
- arr[i] = c; نحط الحرف c في كل عنصر من المصفوفة.

مثال:
لو size = 5 و c = 'A'، راح تصير المصفوفة كذا:

العنوان (Address)	المحتوى (Value)
0x100	'A'
0x101	'A'
0x102	'A'
0x103	'A'
0x104	'A'

إرجاع المصفوفة

return (arr);

- بعد ما خلصنا تعبئة المصفوفة، نرجع **المؤشر arr** اللي يشير لأول عنصر في المصفوفة.
- الحين اللي استدعى الدالة (main.c) يقدر يستخدم المصفوفة اللي سويناها.

توضيحات :

الذاكرة (Memory Layout) بعد استدعاء `create_array(5, 'A')`

العنوان (Address)	المحتوى (Value)	حجم العنصر (Size)
0x100	'A'	1 بايت char
0x101	'A'	1 بايت char
0x102	'A'	1 بايت char
0x103	'A'	1 بايت char
0x104	'A'	1 بايت char

- كل خانة في الجدول تمثل مكان في الذاكرة العنوان
- كل مكان حجمه 1بايت لأنه من نوع char.
- الدالة malloc تحجز أماكن متجاورة في الذاكرة .
- كل مكان في المصفوفة يحتوي على الحرف 'A'.
- العناوين في الذاكرة تكتب بالنظام السادس عشري (Hex) عشان تكون أوضح .
- `malloc` يحجز مكان متغير في الذاكرة، لكنه دائمًا يكون متجاور .
- كل `char` يأخذ 1بايت، لذلك العناوين تزيد بـ `1`.
- لو كان النوع `int` أو `double`، العناوين تقفز بقيم أكبر `4` أو `8`