swim lesson : 27/02/2025

How To Login During Exam

Step 1: Login
User: exam
Password: exam

Step 2: Terminal
Type “examshell”

Step 3: Login
<yourintrausername>
<your password>

Step 4: The exam starts
Name your folder after the *exercise* needed
Name your file in the folder as the *exercisename.c*

If there are any questions you have:

Write it down!

Swimming Lesson Objectives Today:
A: Structure*
B: Variable and Data Types*
C: Input and Output*
D: Operators
E: Control Flow (while and if)
Swimming Lesson Objectives Next Wednesday:
F: Function
G: Arrays
H: Strings
I: Pointers

Welcome to Swim Lesson #2

In this lesson: We have 3 levels and a bonus question.

First answer the bonus question gets you a teenie present :D

LEVEL: ONE

- *Structure: Calling a function by its name*
- *Variable and Data Types: Using %s and declaring/assigning*

	Creating "Techno" with printf()	Creating "Techno" with write()	Notes:

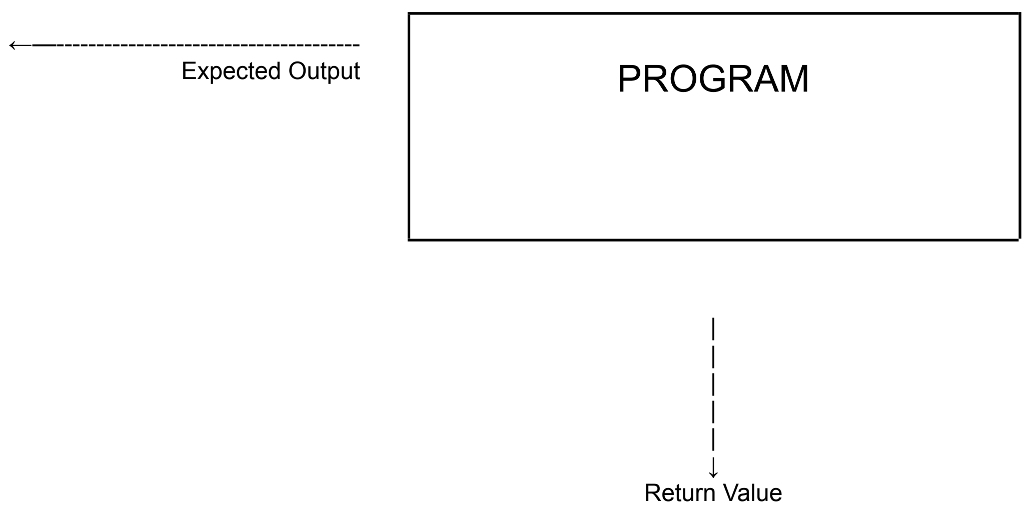
[Function]	`#include <stdio.h> library`
	`void ft_music(void)`
	`{`
	`    char t[] = "Techno";`
	`    printf("%s",  t);`
	`}`
		`#include <unistd.h> library`
		`void ft_music(void)`
		`{`
		`    write(1, "Techno", 6);`
		`}`
			`#include <library.h>`
			
			Without including the specified library, you won't be able to use the function.

[Program]	`int main(void)`
	`{`
	`    ft_music(    ); calling`
	`    return (0);`
	`}`
		`int main(void)`
		`{`
		`    ft_music(    ); calling`
		`    return (0);`
		`}`
			`returntype main (parameters)`
			`{`
			`    ft_name(parameters);`
			`    return (returnvalue);`
			`}`

[Compiling]	`gcc -Wall -Wextra -Werror <filename.c>`		cc - The compiler program
[Output File]	a.out	a.out	To see the file: ./a.out
[Expected Output]	Techno	Techno	The purpose of our code was to create the string "Techno"
[Return Value]	0	0	Typically, return (0); indicates successful execution.

Notes:

- When you create a function with a return type in C, you get two things: <u>the return value and any additional output the program may produce (such as printed messages).</u>
- There are ways to use both (you'll see in Level 3).



LEVEL TWO:

- *Control Flow: while() versus if()*
- *Variable and Data Types: Using %d and declaring/assigning*
- *Pointer Arithmetic*

While loops *occur until the condition created is met*	***If loops*** *occurs in the instance it happens*

while loops

[Program]

```
#include <stdio.h>

int main(void)
{
    int i = 0;

    while (i < 5)
   {
printf("Number: %d", i);
        i++;
   }
    return (0);
}
```

if loops

[Program]

```
#include <stdio.h>

int main(void)
{
    int i = 0;

    while (i < 10)
   {
     if (i % 2 == 0)
     {
printf("Even Index: %d", i);
     }
        i++;
     }
    return (0);
}
```

MEANING OF OPERATORS

```
a / b = division
a % b = remainder of division
a * b = multiplication
a - b = subtraction
a + b = addition
```

`while (i < 5)`
So as i = 0, it will keep the loop going until i = 4 because our condition is that i cannot be more than 5.

`if (i % 2 == 0)`
This means when the number is divided by two and leaves NO REMAINDER

[Compiling]

```
gcc -Wall -Wextra -Werror <filename.c>
```

	while loops	if loops
[Output File]	a.out	a.out
[Expected Output]	Number: 01234	Even Index: 02468
[Return Value]	0	0

Exploration Tips:
*A: There is a way to traverse the string using **pointer arithmetic**.*
*B: There is the concept of **decrementer** explore how you can utilise that and what conditions you need.*

Traversing a string Using i++;

[Program]

```
#include <unistd.h>
int main(void)
{
int i = ‘0’;
char str[] = "0123456789";

while (str[i] <= ‘5’)
  {
   write (1, &str[i], 1);
   i++;
  }
return (0);
]
```

Character Traversal Using Assignment

[Program]

```
#include <unistd.h>
int main(void)
{
char a = ‘a’;

while (a <= ‘g’)
  {
   write (1, &a, 1);
   a = a + 1;
  }
return (0);
]
```

Notes:

You can use `i++;` as an incrementer

OR

You can use also an **assignment**

`a = a + 1;`
Other ways to write the assignment
`a += 1`

[Compiling]

```
gcc -Wall -Wextra -Werror <filename.c>
```

	Traversing a string Using i++;	Character Traversal Using Assignment
[Output File]	a.out	a.out
[Expected Output]	012345	abcdefg
[Return Value]	0	0

LEVEL: THREE

- *Using a return value in an if statement*

[Function]

A

```
#include <stdio.h>

int ft_numbers(char *str)
{
   int i = 0;
   while (str[i] != ‘\0’)
   {
      if (str[i] >= ‘0’ && str[i] <= ‘9’)
      {
         return (1);
      }
      i++;
   }
   return (0);
}
```

B

```
#include <stdio.h>

int ft_numbers(char *str)
{
   int i = 0;
   while (str[i] != ‘\0’)
   {
      if (str[i] >= ‘0’ && str[i] <= ‘9’)
      {
         return (1);
      }
      else
         return (0);
   }
   return (0);
}
```

[Program]

A

```
int main(void)
{
char msg[] = “Numbers exist”;
char a[] = “Hello”;
char b[] = “Hell0”;

   if(ft_numbers(b) != 0)
   {
      printf(“%s”, msg);
   }
   else
      printf(“%s”, b)
   return (0)
}
```

B

```
int main(void)
{
char msg[] = “Numbers exist”;
char a[] = “Hello”;
char b[] = “Hell0”;

   if(ft_numbers(b) != 0)
   {
      printf(“%s”, msg);
   }
   else
      printf(“%s”, b)
   return (0)
}
```

[Compiling]

```
gcc -Wall -Wextra -Werror <filename.c>
```

	A	B
[Output File]	a.out	a.out
[Expected Output]	?????	?????
[Return Value]	?????	?????

Exploration tips:
Look up return-type **char** and try to apply the concept in your code.

BONUS QUESTION: POINTER QUESTION

REWARD: Candy Bar
Hint: It’s a one line function

ft_add: Create a function that adds the value of the second integer to the pointer to the first integer.

prototype: void   ft_add(int *ptr, int n);

```
void        ft_add(int *ptr, int n)
{

}
```