



Exercise Set — Parameter Passing in C

Focus: Pass-by-Value vs Pass-by-Reference

Section A — Multiple Choice Questions (MCQs)

1. What happens when a function parameter is passed **by value**?
 - a) The function can modify the original variable.
 - b) The function works with a copy of the variable.
 - c) The function directly accesses the original memory location.
 - d) The function allocates memory dynamically for the variable.
2. Which statement is **TRUE** about passing a parameter **by reference** in C?
 - a) It is achieved using the `&` operator in the function call and `*` operator inside the function.
 - b) It creates a duplicate variable in memory.
 - c) It prevents the function from modifying the variable's value.
 - d) It is only possible with arrays, not with integers.

3. If a function prototype is written as:

```
void change(int *x);
```

and called as:

```
change(&n);
```

this demonstrates:

- a) Pass-by-value
 - b) Pass-by-reference
 - c) Type casting
 - d) Memory allocation
-

Section B — Debug the Wrong Code

Code:

```
#include <stdio.h>

void update(int a) {
    a = a + 10;
}

int main() {
    int num = 5;
    update(&num);
    printf("Updated value: %d", num);
    return 0;
}
```

Tasks:

- Identify and fix **all mistakes** so that the `update()` function **changes** the value of `num` in `main()`.
-

Section C — Predict the Output

1. Code Snippet 1:

```
#include <stdio.h>

void modify(int a) {
    a = a * 2;
}

int main() {
    int num = 10;
    modify(num);
    printf("%d", num);
    return 0;
}
```

Question: What will be printed? Explain **why**.

2. Code Snippet 2:

```
#include <stdio.h>

void modify(int *a) {
    *a = *a * 2;
}

int main() {
    int num = 10;
    modify(&num);
    printf("%d", num);
    return 0;
}
```

Question: What will be printed? Explain **why**.

Section D — Program Writing Task

Problem Statement:

You are building a mini RPG (Role Playing Game) stat management system for a character. The player has three main attributes:

- **Health Points (HP)**
- **Mana Points (MP)**
- **Experience Points (XP)**

Write a program that:

Function 1: LevelUpByValue



- Takes HP, MP, and XP **by value**.
 - Increases HP by 10%, MP by 5%, and XP by 20.
 - Displays the updated values **inside the function**.
-

Function 2: LevelUpByReference

- Takes HP, MP, and XP **by reference**.
 - Applies the same increases.
 - Displays the updated values **inside the function**.
-

In `main()`:

1. Initialize the player's stats:

HP = 100, MP = 50, XP = 40
 2. Display the **original stats**.
 3. Call `LevelUpByValue` and display stats after returning to `main()`.
 4. Call `LevelUpByReference` and display stats after returning to `main()`.
 5. Clearly highlight in output that **only pass-by-reference actually changes stats in `main()`**.
 6. Add logic to check if the player qualifies for a **Level Upgrade**:
 - Level up if **XP >= 50** after all updates.
 - Display " Player leveled up!" or " Player needs more XP" accordingly.
-

Bonus Challenge:

- Ask the user to input the **bonus percentage** for HP and MP and XP increment before calling each function.
- Ensure that **the first function (by value)** uses the user's input but still doesn't update `main()` values permanently.