Task (1) – Ibrahim Mohamed

Link Codeforces

https://codeforces.com/submissions/Ibra7im_Mo

1. `while` Loop

- **Purpose**: Repeats a block of code as long as the condition is true
- **Execution**: Checks condition first, then executes the code block
- **Best for**: Situations where you don't know the exact number of iterations needed

```
while(condition) {
    // Code to execute
}
```

Example:

```
int i = 1;
while(i <= 5) {
    printf("%d ", i);
    i++;
}
// Output: 1 2 3 4 5
```

---

2. `do-while` Loop

- **Purpose**: Executes the code block first, then checks the condition
- **Execution**: Always runs at least once
- **Best for**: When you need to execute code before checking the condition

```
do {
    // Code to execute
} while(condition);
```

Example:

```
int i = 1;
do {
    printf("%d ", i);
    i++;
} while(i <= 5);
// Output: 1 2 3 4 5
```

---

3. `for` Loop

- **Purpose**: Repeats code a specific number of times
- **Execution**: Combines initialization, condition, and update in one line
- **Best for**: When you know exactly how many times to repeat

```
for(initialization; condition; update) {
    // Code to execute
}
```

Example:

```
for(int i = 1; i <= 5; i++) {
    printf("%d ", i);
}
// Output: 1 2 3 4 5
```

Loop Comparison

Loop Type	Key Difference	When to Use
`while`	Checks condition first	When iteration count is unknown
`do-while`	Executes at least once	When you need initial execution
`for`	All controls in one place	When iteration count is known

Important Notes

1. **Avoid infinite loops**: Ensure the condition will eventually become false
2. **`break` statement**: Immediately exits the loop
3. **`continue` statement**: Skips current iteration and moves to next one
4. **Nested loops**: You can put loops inside other loops

Example of `break` and `continue`:

```
for(int i = 1; i <= 10; i++) {
    if(i == 5) break;      // Exits loop when i=5
    if(i % 2 == 0) continue; // Skips even numbers
    printf("%d ", i);
}
// Output: 1 3
```

Special Cases

- **Infinite loops**: Can be created intentionally with `while(1)` or `for(;;)`
- **Empty loops**: Loops with no body (just a semicolon)
- **Loop variables**: Can be declared inside `for` loops (C99+)

Infinite loop example:

```
while(1) {
    // Code that runs forever
    // Needs break statement to exit
}
```

Summary

- **`while`**: "Repeat while condition is true"
- **`do-while`**: "Execute once, then repeat if condition is true"
- **`for`**: "Repeat a specific number of times"