

Question: I am compiling the file with gcc and then i try to run a.exe file that gcc is generating, but it does not print anything on the screen. Why?

Answer: The reason your program does not print anything is likely a **Stack Overflow**.

The Cause

In the C code provided, the LL1Analyzer struct is extremely large.

- It contains TableEntry table[MAX_PRODS * MAX_SYMBOLS].
- With MAX_PRODS=200 and MAX_SYMBOLS=100, that is **20,000 entries**.
- Each entry is about 104 bytes.
- Total size \approx **2 MB**.

On Windows (and many default GCC setups), the **stack size limit is often 1 MB**. Because you declared these variables inside main (LL1Analyzer ana_sem;), the program tries to reserve 2 MB of stack memory immediately upon starting. This fails, causing the program to crash silently before it even reaches the first printf.

The Fix

You need to move these large variables off the stack. The easiest way is to make them **static** (or move them outside main as global variables).

Here is the corrected main function. Replace your existing main with this one:

C

```
// ... (rest of the code remains the same)

int main() {
    // -----
    // REQUIREMENT 1: Seminar Grammar
    // -----
    printf("\n=====\\n");
    printf(" REQUIREMENT 1: SEMINAR GRAMMAR & SEQUENCE\\n");
    printf("=====\\n");

    // FIX: Use 'static' to store these in the data segment instead of the stack
    static Grammar g_sem;
    static LL1Analyzer ana_sem;

    ana_sem.g = &g_sem;
```

```

load_grammar(&g_sem, "seminar_grammar.txt");
build_first(&ana_sem);
build_follow(&ana_sem);
build_table(&ana_sem);

if (ana_sem.conflict_flag) {
    printf("[FAIL] Seminar Grammar is not LL(1).\n");
} else {
    printf("[OK] Seminar Grammar is LL(1). Parsing 'seq.txt'...\n");
    int output_band[MAX_STACK];
    int out_len = 0;
    if (parse_sequence(&ana_sem, "seq.txt", output_band, &out_len)) {
        printf("Sequence parsed successfully!\nOutput (Production String): [");
        for(int i=0; i<out_len; i++) printf("%d%s", output_band[i], (i<out_len-1)?", ":"");
        printf("]\n");
    } else {
        printf("Failed to parse seminar sequence.\n");
    }
}

// -----
// REQUIREMENT 2: DSL Grammar
// -----
printf("\n===== \n");
printf(" REQUIREMENT 2: DSL GRAMMAR & PIF\n");
printf("===== \n");

// FIX: Use 'static' here as well
static Grammar g_dsl;
static LL1Analyzer ana_dsl;

ana_dsl.g = &g_dsl;

load_grammar(&g_dsl, "dsl_grammar.txt");
build_first(&ana_dsl);
build_follow(&ana_dsl);
build_table(&ana_dsl);

if (ana_dsl.conflict_flag) {
    printf("[FAIL] DSL Grammar is not LL(1).\n");
} else {
    printf("[OK] DSL Grammar is LL(1). Parsing 'pif.txt'...\n");
}

```

```

    int output_band[MAX_STACK];
    int out_len = 0;
    if (parse_sequence(&ana_dsl, "pif.txt", output_band, &out_len)) {
        printf("PIF parsed successfully!\n");

        // Build Tree
        int pid_idx = 0;
        ParseNode *root = recursive_build(&g_dsl, g_dsl.start_symbol, output_band, &pid_idx,
out_len);
        print_tree_bfs(root);

    } else {
        printf("Failed to parse PIF.\n");
    }
}

return 0;
}

```

Important Checklist

After applying the fix, ensure the following to see output:

1. **Input Files:** Make sure seminar_grammar.txt, seq.txt, dsl_grammar.txt, and pif.txt are in the **same folder** as your a.exe.
2. **Recompile:** Run `gcc ll1.c -o a.exe` again.
3. **Run:** Execute `./a.exe` (or just a.exe in Command Prompt).