📋 Homework-1: Student Data Processor

☑️ Section 1: Assignment Overview

Implement a C program that reads messy student data from an input file, cleans and extracts relevant information, and outputs clean CSV-formatted data to stdout.

☑️ Section 2: Learning Objectives

- File input/output operations in C
- String processing and text parsing
- Data cleaning and validation
- CSV format generation
- Memory management and error handling

☑️ Section 3: Problem Description

You are given a messy input file containing student records with inconsistent formatting, extra whitespace, noise characters, and data scattered across multiple lines. Your task is to extract clean student information and output it in a standardized CSV format.

Data Fields to Extract

Each student record contains the following fields:

- **Name**: First name and last name only (exactly 2 words)
- **PID**: FIU student ID (format: FIU followed by 5 digits)
- **Email**: Student email address
- **Phone**: 10-digit phone number
- **Department**: Academic department
- **Year**: Academic year (Freshman, Sophomore, Junior, Senior, Graduate)

Input Data Anomalies

Your program must handle the following messy data issues:

- ⚠️ **Extra whitespace and tabs** around field values
- ⚠️ **Multiple newlines and blank lines** between records
- ⚠️ **Noise words mixed with real data** (e.g., "xaxsaxa", "garbage", "123")
- ⚠️ **Data scattered across multiple lines**
- ⚠️ **Inconsistent spacing and formatting**
- ⚠️ **Random separator lines** (===, ---, etc.)

Data Assumptions

- ⚠️ **Name has only** `<Firstname Lastname>` (exactly 2 words)
- ⚠️ **PID is exactly** FIU followed by 5 digits (8 chars total)
- ⚠️ **Email is well-formed** with @ symbol
- ⚠️ **Phone number is exactly** 10 digits
- ⚠️ **Department and year are non-empty** after cleaning

☑️ Section 4: Technical Requirements

Reference Files

- 📁 **Input example**: `"Testing/Testcases/input1.txt"`
- 📁 **Expected output**: `"Testing/Expected_Output/output1.txt"`
- 📖 **Refer to these files** for understanding the exact requirements

☑️ Section 5: Provided Files Framework

File Structure

```
PROVIDED_FILES/
├── student_submission.c # Template file (implement your solution here)
├── autograder.sh        # Testing script
└── Testing/
    ├── Testcases/
    |   └── input1.txt  # Sample input data
    └── Expected_Output/
        └── output1.txt # Expected output format
```

☑️ Section 6: Server Testing Requirements

- **Implement your solution** in `student_submission.c`
- **Create your own** `Makefile`
- **Run the autograder**: `./autograder.sh`
- **Check your grade** and fix any issues
- **Repeat until** you achieve the desired score
- ⚠️ **MUST test on ocelot server**: `ocelot-bbhatkal.aul.fiu.edu`
- ⚠️ **Test thoroughly before submission** - no excuses accepted
- ⚠️ **"Works on my computer" is NOT accepted** as an excuse
- ✅ **If you pass all test cases on the server, you will likely pass instructor's final grading**
- ✅ **What you see is what you get** - autograder results predict your final grade

☑ Section 7: Submission Requirements

File Requirements (⚠ exact names require)

- **student_submission.c**: Your complete implementation
- **Makefile**: Build configuration
- **README**: Student information and documentation

`Makefile` Requirements

- ⚠ **Exact filename**: `Makefile` (no extension, exact capitalization)
- ⚠ **Must compile** `student_submission.c` to executable named `run`
- ⚠ **Must include** `clean` target
- ⚠ **Should use** appropriate compiler flags **-std=c17 -Wall -Wextra -Werror -pedantic -g -O0**

`README` Requirements

Your README ⚠ **MUST include**:

```
# Student Information
- Full Name: [Your Full Name]
- PID: [Your FIU Panther ID]
- Section: [Your Course Section]
- FIU Email: [Your @fiu.edu email]

# Assignment: Student Data Processor
[Brief description of your implementation approach]
```

Deliverables `ZIP` File Submission

- ⚠ You are required to submit your work as **a single ZIP archive file**
- ⚠ **Filename Format**: your `Firstname_Lastname.zip` (**exact format required**)
- 📁 **Contents**:
  - `student_submission.c`
  - `Makefile`
  - `README`

```
John_Doe.zip
├── student_submission.c # Your implementation
├── Makefile             # Your Makefile
└── README               # Your details
```

☑️ Section 8: Grading Criteria

🤖 Autograder Testing

- ⚠️ **Your program will be tested** against the provided test case using the autograder
- ⚠️ **Exact output matching required** - any deviation results in point deduction
- ⚠️ **Output must be written to stdout**, not to a file
- ⚠️ **Program must compile** without errors or warnings

⛔ Penalties

- ⚠️ **Your program will be tested** against the provided test case using the autograder
- ⚠️ **Your program will be tested** against the provided test case using the autograder
- ⚠️ **Your program will be tested** against the provided test case using the autograder
- ⚠️ **Missing README.md**: **-10 points**
- ⚠️ **Incorrect ZIP filename**: **ZERO grade** (will not be graded)
- ⚠️ **Missing Makefile**: **ZERO grade** (compilation failure)
- ⚠️ If the **autograder does not run with your Makefile**, your submission will fail the final testing and receive a **ZERO grade** (compilation failure)
- **Wrong source filename**: ⚠️ **ZERO grade** (autograder cannot find file)
- **Late submission**: **NOT ALLOWED**

☑️ Section 11: Academic Integrity

- 👤 **This is an individual assignment**
- 💬 **You may discuss concepts** but not share code
- 📝 **All submitted code** must be your original work
- ⚠️ **Plagiarism** is a serious offense and will result in penalties.

🚀 **Good luck with your homework!**