Week 2 Practice Exercises

Dear students,

You are provided with sample code files for three exercises. **Your tasks are to write Makefiles and debug faulty codes** that meet the specified requirements, **paying special attention to the required compilation flags**. The exercises are designed to test your understanding of Makefile basics, compilation flags, efficient build processes, and debugging in C++.

- **Exercise 1** requires you to debug a pre-written code so it can run without any errors.
- **Exercise 2** focuses on compiling a single source file with the `-std=c++11` flag.
- **Exercise 3** involves compiling multiple source files, using object files, and includes both the `-std=c++11` and `-g` flags.

Be sure to test your Makefiles and codes thoroughly before submission to ensure they meet all the requirements. Good luck!

EXERCISE 1

Objective

Debug given code files so the program can run without any errors, debugging may require you to add or delete lines and change some of the code.

Given

- **Source files**: **`main1.cpp`**, **`GMM.cpp`**.
- **Header files**: **`GMM.h`**.
- **Input files**: **`Product.txt`**.

Requirements

- Code should run without any errors.
- Code should not have any memory leaks.
- Do not make any changes on the input file.
- Do not make any changes to the code between <DO NOT CHANGE> and </DO NOT CHANGE> comments.

Instructions

A gym has a specialized vending machine (Gym Meal Machine i.e. GMM) that allows its members to choose foods not just with slot number but with its nutritional values (protein, calorie, etc.) as well. To ensure a varied selection, the gym wants to provide a diverse range

of calorie options. They have tasked their new intern with the task of writing a code that can check this automatically. The intern has tried to do a good job, but ended up with unknown number of errors. Help the intern debug the code so it can run efficiently and without any errors.

Submission format:

- b<studentID>.zip
    - GMM.h
    - GMM.cpp

EXERCISE 2

Objective

Write a simple Makefile to compile a single C++ source file into an executable using the C++11 standard.

Given

- A single source file: **`main2.cpp`**.

Requirements

- The Makefile should compile `main2.cpp` into an executable named `app`.
- The compilation must use the C++11 standard (flag`-std=c++11`).
- Include a `clean` target to remove the executable.

Instructions

Create a Makefile that compiles `main2.cpp` into an executable named `app` using the `-std=c++11` flag. Ensure that you include a `clean` target in your Makefile to remove the generated executable.

Submission format:

- b<studentID>.zip
    - Makefile

EXERCISE 3

Objective

Create a Makefile to compile a program consisting of multiple source and header files, using the C++11 standard and including debugging information.

Given

- **Source files**: **`main3.cpp`**, **`utils.cpp`**.
- **Header files**: **`utils.h`**.

Requirements

- The Makefile should compile the sources into object files and then link them into an executable named `app`.
- The compilation must use the C++11 standard (`-std=c++11`) and include the debugging flag (`-g`).
- Utilize pattern rules or suffix rules for compilation.
- Include a `clean` target.

Instructions

Create a Makefile that:

- Compiles `main3.cpp` and `utils.cpp` into object files.
- Links the object files into an executable named `app`.
- Uses the `-std=c++11` and `-g` compilation flags.
- Utilizes pattern rules or suffix rules to simplify your Makefile.
- Includes a `clean` target to remove all generated files.

Submission format:

- b<studentID>.zip
    - Makefile