This sheet is an example of using more than one data structure in a single application. You are required to write a program that keeps track of the customers visiting a car workshop (as explained below); the program utilizes two data structures, a stack and a queue to have customers data in particular orders. Before coding the program succeed first in linking the three files, `stack.c`, `queue.c`, and `main.c`; follow the following guide.

**Since the element type of the stack, queue, and the main program is the same, we need to be more structured and organized and define this common element type in a separate file** `global.h`. In this file we should have all the definitions that are common to all of the three modules; these definitions are:

- the maximum stack size
- the maximum queue size
- the `ElementType`, which will be the type of a customer and contains the following data: Name, ID, and model year.
- the `StackEntry`, and `QueueEntry`, which will be exactly the same as `ElementType`.

Now, `stack.c` must include `stack.h` and the latter includes `global.h`; why? Also, `queue.c` must include `queue.h` and the latter includes `global.h`. Finally, `main.c` includes all the three header files; why? However, this will cause a "redefinition" error since the definitions in `global.h` will appear again in the other two included header files because they also include `global.h`. To resolve this problem, we need to start `global.h` by:

```
#ifndef GLOBAL_H
#define GLOBAL_H
```

then end it by

```
#endif
```

These statements are "Preprocessor Commands" that are processed before compilation. More explanation will be provided in labs; you can also refer to your C language text book.

## Main Program

The main program should display the following menu:

1. Add a New Customer.
2. Serve a Customer.
3. How many Customers are waiting?
4. Display Customers Information.
5. Display Customers information in a "most-recent" Order.
6. Exit

**Hints:** By choosing "Add a New Customer" you should enter the data of the new arriving customer and save it such that he has the least priority among others. By choosing "Serve a customer" you should display the data of the first arriving customer then dismiss him from the system. "Display Customers Information" prints on screen the data of the current waiting customers without serving them; this can be done, e.g., by serving all the customers to print out their information then adding them again in the queue. (Of course, this can be done more efficiently and elegantly by coding the function `DisplayQueue`, in the implementation level, that does the same. We will explain this in lectures). "Display Customers in a most-recent Order" can be done by copying the data to a structure that reverses the order.