Some remarks before we start:

- Your program should be in an infinite loop. When an operation is completed, it should display the main screen so that the user can go on perpetually.
- Whenever you make a page access, whether it be a read or write, you should output it to the screen with its page address. You can differentiate between read and write like `Reading page 12`, or `Writing page 7`. You can also select to write `Accessing page 12` regardless of the access type.
- All types should have a primary key field so that retrieval and deletion can be run without problems. For this, you have three options:
  - You assume that the first field of each type is its key. For instance, `Name` for `Cat(Name, Age, Owner)`.
  - During type creation, after the user inputs all field names, you ask which field is the key.
  - You create an ID field for all types without user creating. In that case, a type such as `Cat(Name, Age, Owner)` will be created and used as `Cat(ID, Name, Age, Owner)`

  I would personally choose the first option, but you are free to do as you like.
- Remember that you can read or write a single page at a time.
- The changes the user makes should persist in the system.

Non-compliance to the aforesaid items will result in grade reduction.

You are free to design a user interface as you like. You can make it GUI-based but a CLI interface is enough. We will give some CLI menu examples. These are to give you some ideas, you do not have to copycat them.

We said that there are 7 operations to implement, and divided these into two: record and type operations. An opening screen can look like the following:

> Please select one:
>
> 1. Type operations
> 2. Record operations

You can alternatively say DDL and DML operations for type and record operations, respectively. Names don't really matter.

Then, suppose the user selected Type. The next screen screen can be like the following:

1. Create type
2. Delete type
3. List all types

Supposing the user selected Create, it goes on:

- Enter type name:
  (user input)
- Enter number of fields:
  (user input)
- Enter name for field 1:
  (user input)
- Enter name for field 2:
  (user input)
  ...

Similarly, if the user selected to run a record operation, the screen would be like the following:

- Create record
- Delete record
- Retrieve a record
- List all records of a type

Supposing the user selected to delete a record, you can list the types present in the system:

Select the type of the record you wish to delete:

- Cat
- Dog
- Hamster

Alternatively, you can have the user input the type name as:

Enter the type of the record you wish to delete:
(user input)

Personally, I think the latter one is less user-friendly and more error-prone. However, since we will run the demos with valid data, and the user will be yourself, it does not matter.

After the user selects the type, you should ask which record to delete, as the following:

Select the record to delete:

- 1. Name: Tekir, Age: 2
- 2. Name: Pamuk, Age: 4

Again, you can choose to have the user enter the key field rather than make a selection:

Enter the key of the record you wish to delete:
(user input)

Your design can be totally different from these examples. I have seen people have the user input commands, for instance to create a type:

CreateType Cat Name Age Owner

So that they input all the required information in one line. However, remembering these commands during the demo can be a struggle.