```
#ifndef IIKH_H
#define IIKH_H

#include <iostream>
#include <string>
#include <vector>
#include <map>

using namespace std;

// class for ingredients used in a recipe
class Ingredient {
public:
    // constructor: takes the name, unit, and the quantity of the ingredient as input parameters
    Ingredient(string name, string unit, double quantity);

    /*
     * change_unit: changes the selected ingredient's unit of measurement
     * convert_unit: converts the unit (e.g., g to kg, ml to l)
     */

    void change_unit(string name);
    void convert_unit(string new_unit);

private:
    string name;     // name of the ingredient
    string unit;     // unit of measurement (e.g., "g", "oz", "ml")
    double quantity; // quantity of ingredient
};
```

```
// class for recipes
class Recipe {
public:
   // constructor: takes the name, list of ingredients, cooking instructions, a
mount of servings and preparation time as input parameters
   Recipe(string recipe_name, vector<Ingredient> ingredients, string instructi
ons, int servings, int prep_time);


   /*
    * add_ingredient: adds an ingredient to the recipe
    * remove_ingredient: removes an ingredient from the recipe
    * update_ingredient_quantity: modifies the quantity of an ingredient
    * set_instructions: sets the instructions for the recipe
    * display_recipe: displays a recipe with pictures and instructions included
    * get_instructions: returns the instructions for the recipe
    * get_recipe_name: returns the name of the recipe (food name)
    * get_ingredient: returns a list of ingredients of the recipe
    */


    void add_ingredient(Ingredient ingredient);
   void remove_ingredient(string ingredient_name);
   void update_ingredient_quantity(string ingredient_name, double quantity);
   void set_instructions(string instructions);
   void display_recipe(string name);
   vector<Ingredient> get_ingredients();
   string get_recipe();
   string get_recipe_name();


private:
   int servings;                    // number of servings
   int prep_time;                   // preparation time for the recipe
   string recipe_name;              // name of the recipe (food name)
   string instructions;             // set of instructions for the recipe
   vector<Ingredient> ingredients; // a list of ingredients in the recipe
};
```

```
// class for a meal plan
class MealPlan {
public:
   //constructor: takes name of the meal plan, type of the plan, and a list of r
ecipes in the meal plan as input parameters
   MealPlan(string plan_name, string type, vector<Recipe> recipes);

   /*
    * add_recipe: adds a recipe to the meal plan
    * remove_recipe: removes a recipe from the meal plan
    * update_recipe: update recipe details in a meal plan
    * display_meal_plan: searches for the meal plan from the database and retur
ns the meal plan and the grocery list.
    * get_recipes: returns recipes of the meal plan
    * get_grocery_list: returns a grocery list for a meal plan
    */

   void add_recipe(Recipe recipe, string type);
   void remove_recipe(string recipe_name, string type);
   void update_recipe(string recipe_name, Recipe new_recipe);
   void display_meal_plan(string plan_name);
   vector<Recipe> get_recipes(string plan_name);
   vector<Ingredient> get_grocery_list(string plan_name);

private:
   string type;              // type of the meal plan (e.g., "weekly", "lunch")
   string plan_name;         // name of the meal plan (e.g., "John's Meal Plan")
   vector<Recipe> recipes;  // list of recipes in the meal plan
};
```

```
// class for the database of recipes and meal plans
class RecipeDB {
public:
    //constructor: loads itself upon creation
    RecipeDB();


    /*
     * add_recipe: adds a recipe to the database
     * remove_recipe: removes a recipe from the database
     * sort_recipe: sorts recipes according to the option (e.g., "ascending", "d
escending", "category")
     * display_all_recipes: displays all recipes in the database
     * list_meal_plans: lists all available meal plans
     * save_to_file: saves the current database to a file
     * load_from_file: loads the last save from a file
     * search_recipe: returns a specified recipe
     */


    void add_recipe(Recipe recipe);
    void remove_recipe(string recipe_name);
    void sort_recipe(string option);
    void display_all_recipes();
    void list_meal_plans();
    void save_to_file();
    void load_from_file();
    Recipe search_recipe(string recipe_name);


private:
    vector<Recipe> recipes;  // collection of recipes in the database
    string sort_option;      // sorting option (e.g., "ascending", "descending")
};
```

```
// class for the display(UI) of the IIKH system
class IIKHDisplay {
public:
    // constructor: opens a new window and displays a welcome message
    IIKHDisplay();


    /*
     * display_title_screen: displays a welcome messages and a start button
     * display_main_menu: displays the main menus
     * display_recipes_menu: displays the recipes menu along with available sele
ctions
     * display_plans_menu: displays the meal plans menu along with available sel
ections
     */


    void display_title_screen()
    void display_main_menu();
    void display_recipes_menu();
    void display_plans_menu();
};


#endif
```