1 Project Objective

In this group project, you must work in a team to complete the development of a C++ program to implement a Restaurant Order Management System (ROMS). This program implements a simple “character oriented” menu-driven user interface.

2 Restaurant Order Management System

The restaurant menu is organized by categories (appetizers, soups, salads, entrees, sides and drinks) of menu items. Each menu item has a name (e.g., nasi goreng, or fried rice), price and associated recipe. A recipe for a menu item has a chef, preparation instructions and associated ingredients. The ingredients are identified by their ingredient_id and the quantity of the ingredient needed to prepare a particular recipe, the unit of measure and a name.

Customers enter and are seated at tables. An order has an order_id, server_id, associated table_id, date and time. Entries on an order have an associated menu_item_id, order_id, seat_id, and order quantity (usually one). All ID fields are at least four digit numbers (but seat IDs are A thru H). All name fields are a single word. A date field is three numbers: year month day separated by a space. A time value is two numbers: hour (0–23) and minute separated by a space. Description and instruction fields consist of one or more words that terminate with a # sign (thus read characters until you encounter the # sign).

Each of the above types has an associated .h file and .cpp files, which are provided initially in skeletal form. Initial code is also provided for a skeletal interactive, menu-driven user interface. It consists of a C++ main.cpp program and a user_interface type. Three sample restaurant data files are also provided for you to use in development and testing. Format of data files is (N is the number of records of a given type):

Filename: catmenu.dat

    N Category_ID Cat_name . . .
    N Menu_item_ID Category_ID RecipeID Menu_item_name
        Menu_item_price Description # . . .

Filename: recipes.dat

    N Ingredient_ID Recipe_ID Amount_required Units Name . . .
    N Recipe_ID Chef_name Instructions # . . .

Filename: orders.dat
3 Initial Build Step

Each person should independently establish a build environment for the ROMS application. From the course website, download a zip file named ROMS.zip, which contains the source code for establishing a build project under Unix/Linux or Visual Studio C++. First, make sure each person can build, link and run the initial program without making any modifications to the source code. This is your starting point. **To get credit on this step, show your teaching assistant (TA) during lab that you can build and run the program on or before Friday, 21 Oct, 2011.** Individually complete the Pre-Project Questionnaire at this point and turn it in. Your team will then complete the program in a series of increasingly difficult steps, which are described as below.

4 Solving Individual Problems As a Team

**Step A.** Communicate and decide as a team which problem you will do in set A (given below). Then individually work your selected problem. Make changes to the original source code. Add additional comments on each line you add or change that includes your initials and the problem solved (A.1, A.2, or A.3). Comment out any lines you delete (don’t just delete them). Test your code. Demonstrate your solution to your teammates and to your TA. Individually, submit your source code electronically on CSNET.

After all three sub-problems are submitted, communicate and work as a team to merge your solutions into a single source code (which will be used in the next step). **Turn in an interim team report (see description below) and electronic submission of your merged team source code on CSNET by 31 Oct, 2011, 23:59.** Please turn in the cover sheet and a hard copy of your interim team report to your TA. Please complete and turn in the Assignment Satisfaction questionnaire at this point. Your TA will then check to see that all three problems are solved in the merged code and apply the grading rubric.

A.1 When exit is selected, output a message and confirm that the user did not accidentally select exit. Create an Instructions type to be used in the Recipe type. Complete the read() function in Menu.cpp by reading and storing the recipe and ingredients data file.

A.2 The program should output its own identification information before displaying the initial main menu. Add the Date type from Chapter9 to the project for use in the Order type. Add a Time type. Complete the read() function in Menu.cpp by reading and storing the order and order item data.

A.3 Complete the display_main_menu() and display_info() for the Menu.cpp type. Create a Description type to be used in the Menu_Item type. Complete the read() function in Menu.cpp by reading and storing the category and menu item data.
Step B. Communicate within your team and make the choice of who will do what problem below. The team should assign a unique Show, Find and Update query to each member. Write code to complete the appropriate functions in the Menu class. Prompt a user to obtain the desired input for a query.

B.1 Show queries formatted as:
   a. print a menu item recipe [print name, description, ingredients, instructions]
   b. print all menu items sorted by Chef [print item name, ID and Price]
   c. print all menu items in a given category sorted by price [item name, price and description]

B.2 Find queries formatted as:
   a. Display total sales for a given table
   b. Display total sales for a given server
   c. Display total sales for a given menu item

B.3 Update queries formatted as (do not write the new data back to disk yet–see step D.1 later):
   a. Add a new order item to an existing order
   b. Add a new menu item using an existing recipe to an existing category
   c. Add new categories Soups, Salads and Sides

As before, each of you should work your own problem. Start from the merged code turned in for the previous step. Add comments that include your initials, B.1, B.2, B.3, etc. Demonstrate your solution to your teammates and to your TA. **Turn in your individual code on CSNET by 7 Nov, 2011, 23:59. For individual code submission, no late submission will be accepted.**

Then work as a team to merge your solutions into a single source for turn in. Test to make sure all the solutions are in the combined source! **Turn in the merged code of your team that includes the team report on CSNET by 10 Nov, 2011, 23:59.** Use the cover sheet for your hard copy of your team code. Please complete the Team Evaluation form and the Assignment Satisfaction Questionnaire for step B. **Turn in the hard copy of the team code, individually completed team evaluation form and the questionnaire in the beginning of class on Friday, 11 Nov, 2011.**

Step C. Make the choice of who will do what as a team! Work the following query problems.

C.1 a. list all order IDs and the tab price sorted by date [print ID, date and tab price for each order]
   b. list all menu item in a given order by price [print name and price based on quantity]

C.2 c. Display the top five selling menu items by revenue regardless of category
   d. Display the order with the largest tab price

C.3 e. Add a new order given a server, table, date and time
f. Add a new menu item to an existing category

You know what to do by now, but the problems are harder. Start from the merged code turned in on Step B. Submit your individual problem source code on CSNET by Friday, 18 Nov, 2011, 23:59. For individual code submission, no late submission will be accepted.

Turn in your teams merged source code that includes the team report on CSNET by Tuesday, 22 Nov, 2011, 23:59. Please individually complete the Team Evaluation form and the Assignment Satisfaction Questionnaire for step C, and turn these forms and a hard copy of your team code in to your TA before 12:34 p.m. on Wednesday, 23 Nov, 2011.

5 Team Problems (Team to do D.1 and either D.2 or D.3)

D.1 (Required) In the merged code, modify the Exit code added to main() to call a new Menu function you add that writes out updated data files to reflect the new state of the data after a session.

D.2 In the merged code, error check the data read in step B.3 to make sure no order item refers to a non-existent menu item, no menu item refers to a non-existent category, and no recipe refers to a non-existent ingredient.

D.3 In the merged code, verify category, order, recipe and menu item IDs are unique.

Work from a single source (from step C) and label the source code changes you make using your initials and the problem solved–D.1, D.2 or D.3. Test your code and demonstrate the final results to your TA. Turn in a hard copy of your final merged code that will run on Unix, and electronic submission of your team source code with the team report on CSNET by Wednesday, 30 Nov, 2011, 23:59. Please complete the Team Evaluation form and the Assignment Satisfaction Questionnaire for step D and also complete the Post-Project Questionnaire. Please turn in all those hard copies (three individually completed forms and a hard copy of the final team code) to your TA before 5 p.m. on Friday, 2 Dec, 2011.

6 Summary of what you turn in and when

1. Build Step: Show your TA on or before Friday, 21 Oct, 2011 that you can build the ROMS code to get credit for this step. Turn in the Pre-Project Questionnaire.

2. Step A: Turn in your individual source code on CSNET (no individual report in the submission). Turn in your merged team source code with a team report on CSNET. Turn in a hard copy of your team report to your TA. Turn in the Assignment Satisfaction questionnaire. Do all this by 31 Oct, 2011, 23:59.

3. Step B: Turn in your individual code on CSNET by 7 Nov, 2011, 23:59. For individual code submission, no late submission will be accepted.
Turn in the merged code of your team that includes the team report on CSNET by 10 Nov, 2011, 23:59.

Turn in the hard copy of the team code, individually completed team evaluation form and the Assignment Satisfaction questionnaire in the beginning of class on Friday, 11 Nov, 2011.

4. Step C: Submit your individual problem source code on CSNET by Friday, 18 Nov, 2011, 23:59. For individual code submission, no late submission will be accepted.

Turn in your teams merged source code that includes the team report on CSNET by Tuesday, 22 Nov, 2011, 23:59.

Turn in the hard copy of the team code, individually completed team evaluation form and the Assignment Satisfaction questionnaire to your TA before 12:34 p.m. on Wednesday, 23 Nov, 2011.

5. Step D: Turn in your final team source code with the team report on CSNET by Wednesday, 30 Nov, 2011, 23:59. Turn in a hard copy of the final team code, the Team Evaluation form, the Post-Project Questionnaire and the Assignment Satisfaction Questionnaire for Step D to your TA before 5 p.m. on Friday, 2 Dec, 2011.

7 Sample Data Files

A sample of all three input files is included in the ROMS.zip file: catmenu.dat, recipes.dat, orders.dat (Note: .dat files open with Wordpad under Windows.)
Build project (20 points)
Failure to get ROMS to compile: $−10$
Failure to get ROMS to link: $−10$
Failure to get ROMS to execute after it is built: $−10$
(Note: at most 20 points will be deducted if the subtractions add up to more than 20).

Step A. Individual Problems (20 points)
Failure to correctly implement one problem: $−15$
Failure to turn in individual source code: $−10$
Failure to document individual changes in source code: $−5$
Failure to compile: $−10$, or link: $−10$, or execute: $−10$ without terminating abnormally
(Note: at most 20 points will be deducted if the subtractions add up to more than 20).

Step B. Individual Problems (30 points)
Failure to correctly implement one problem: $−20$
Failure to turn in individual source code: $−15$
Failure to document individual changes in source code: $−10$
Failure to compile: $−15$, or to link: $−15$, or run: $−15$ without errors
(Note: at most 30 points will be deducted if the subtractions add up to more than 30).

Step C. Individual Problems (30 points)
Failure to correctly implement one problem: $−20$
Failure to turn in individual source code: $−20$
Failure to document individual changes in source code: $−15$
Failure to compile: $−20$, or to link: $−20$, or run: $−20$ without errors
(Note: at most 30 points will be deducted if the subtractions add up to more than 30).

Individual Project Grade: _____________ /100

Overall Project Grade is the sum of Individual and Team Grades:

____________________ / (100 + 100)
Step A. Individual Problems (20 points)
Failure to implement all three problems: −10
Failure to turn in merged source code: −10
Failure to turn in interim team report containing all sections: −10
Failure to document changes to merged source code: −10
Failure of merged source code to compile without errors: −15, or link: −15, or run: −15
(Note: at most 20 points will be deducted if the subtractions add up to more than 20).

Step B. Individual Problems (20 points)
Failure to implement all three problems: −10
Failure to turn in merged source code: −10
Failure to turn in interim team report containing all sections: −10
Failure to document changes to merged source code: −10
Failure of merged source code to compile without errors: −10, or link: −10, or run: −10
(Note: at most 20 points will be deducted if the subtractions add up to more than 20).

Step C. Individual Problems (30 points)
Failure to correctly implement all three problems: −20
Failure to turn in merged source code: −20
Failure to turn in interim team report containing all sections: −10
Failure to document changes in merged source code: −10
Failure of merged source code to compile: −20, or to link: −20, or run: −20
(Note: at most 30 points will be deducted if the subtractions add up to more than 30).

Step D. Team Problem (30 points)
Failure to correctly implement required part: −15
Failure to correctly implement any of the selective problems: −15
Failure to turn in final team report containing all sections: −20
Failure to document individual changes in final source code: −10 per missing member
Failure of source code to compile: −20, or to link: −20, or run: −20
(Note: at most 30 points will be deducted if the subtractions add up to more than 30).

Team Project Grade: ____________ /100