Spring 2013 Course Syllabus

(Let us go Green - No paper copies, please! – find the latest version on eLearning)

CS 2335.001 – Computer Science II for non-majors

MW 4 pm - 5:15 am ECSS 2.120

Instructor: Kamran Z. Khan

Office: ECSS 4.611

Office Phone: 214-280-7124 (you can call me during office hours)

E-Mail: kkhan@utdallas.edu

Website: elearning.utdallas.edu (lecture notes, assignments, discussions, grades, etc.)

Office Hours: Tuesdays & Thursdays 9:30 am – 11:30 am

By appointment only - email me!

TAs: TBD

Course Pre-requisites, Co-requisites, and/or Other Restrictions

Computer Science I (CS 1335/1337)

Course Description

CS 2335 - Computer Science II for Non-majors (3 semester hours) Exceptions and number formatting. File input/output using Stream classes. Implementation of primitive data structures, including linked lists, stacks, queues, and binary trees. Advanced data manipulation using core classes. This class cannot be used to fulfill degree requirements for majors in the School of Engineering and Computer Science. Students who have taken CE/CS/TE 2336 cannot receive credit for this course. Prerequisite: CS 1335 or CE/CS/TE 1337. (3-0) S

Student Learning Objectives/Outcomes

- Ability to implement a comprehensive OO application
- Ability to create and use primitive data structures
- Ability to use core Java data structures – stack, queue, tree
- Ability to use core Java data structures – lists
- Ability to use core Java data structures – maps
- Ability to implement a GUI for user interaction
- Ability to create and use exception handlers
- Ability to create and use graphical error messages
- Ability to use file input/output – text files
- Ability to use file input/output – object files

Required Textbooks and Materials

Gaddis, Starting Out with C++ From Control Structures through Objects 7/e 0321545885 (My Programming Lab - not needed, 6th edition ok too)

Note: CS dept has made one-time exception and has allowed the instructors to teach this course in C++ only for Spring 2013. This course will revert back to Java (to match CS 2336) in future terms.

Suggested Course Materials

C++ language tutorial http://www.cplusplus.com/files/tutorial.pdf

C++ tutorial http://www.learncpp.com/

C++ reference: http://cppreference.com

Academic Calendar: See "Coures Homepage" witnin elearning for the detailed schedule. It will be updated with lecture notes as the semester proceeds.

Grading Policy

Course credit is only given for work assigned in the course schedule. No extra work will be assigned nor will extra credit be given for any extra work performed by a student. The final grade will be computed as follows:

Tests	60%	Test 1: 15%, Test 2: 20% and Test 3: 25%

Assingments	25%	5 assignments contributing 5% each
Online Participation	10%	1 meaningful post is required in weekly discussion forums.
Class Participation/Attendance	5%	Weekly activity will be posted by Monday & will be due on Sunday. It may be an online quiz, programming exercise or something else.

Letter grades will be assigned as follows:

98-100 A+	92-97 A	90-91 A-
88-89 B+	82-87 B	80-81 B-
78-79 C+	72-77 C	70-71 C-
68-69 D+	62-67 D	60-61 D-
Below 60 F		

Weitghted total in your gradebook shows the current weighted grade based on your graded work. For example, if you have done only test 1, 2 programming assignments, 3 weeks online participation so far, current grade will be based on only those entries.

Course & Instructor Policies

All exams are closed book and closed notes. Exams will focus more on concepts and less on details. Necessary documentation will be provided to avoid the need for memorization as much as possible. All make-up exams are scheduled during the week following the actual exam date at the discretion of the instructor. Make-up exams are only given to those students who coordinate the missing of an exam prior to the originally scheduled exam date.

Instructor is responsible for grading all the tests & weekly participation. TA will be responsible for grading assignments and weekly activities. So, contact the TA directly for any grading related discrepancies for programs. If you cannot resolve it with TA, bring it to instructor's attention.

Students are expected to engage in active learning. You need to post at least 1 meaningful post in weekly discussion forums. It can be a good question, meaningful response to another student's question, interesting observation or anything that adds value to the course. You are also expected to complete a weekly activity every week - it will be posted on Monday every week and it will be due on Sunday. Note that all weekly activities may NOT be graded for correctness - a meaningful attempt alone may fetch full points. Late submissions will receive 0 for class participation and weekly activities, except for serious medical conditions.

In addition to meeting the instructor before or after the class, you can also visit the instructor during respective office hours. You can call instructor's office phone during office hours as well. However, be prepared to hold and wait if the instructor is busy with another student in the office. Additionally, you are welcome to email the instructor or grader within elearning. This is preferred approach specifically if you run into project related issues & you need help to progress. In such scenarios, in addition to problem description & applicable error messages, zip all your source files and include it with your email too, so that we can help you efficiently.

Any standard C++ compiler and Integrated Development Environment (IDE) can be used to develop, debug and run your programs. Code::Blocks, Microsoft Visual Studio, Microsoft Visual C++ Express, NetBeans, Eclipse and jGRASP are a few popular tools. More information about these tools will be provided in elearning.

Here are a few assignments from another course & they are given for reference only. Official ones will be posted under Assignments within elearning.

Assignment #1: C++ program to do grocery checkout.

Write a C++ program to perform grocery check-out procedure for a simple store with max 100 products. Design & use a C++ product class in your program.

When the program starts, it should read the product information file inventory.txt - it contains product information (PLU code, product name, product sales type, price per pound or price per unit & current inventory level) - one product in each line. Here is the sample contents for inventory.txt:

```
4101 BRAEBURN_REG 1 0.99 101.5
4021 DELICIOUS_GDN_REG 1 0.89 94.2
4020 DELICIOUS_GLDN_LG 1 1.09 84.2
4015 DELICIOUS_RED_REG 1 1.19 75.3
4016 DELICIOUS_RED_LG 1 1.29 45.6
4167 DELICIOUS_RED_SM 1 0.89 35.4
4124 EMPIRE 1 1.14 145.2
4129 FUJI_REG 1 1.05 154.5
4131 FUJI_X-LGE 1 1.25 164.1
4135 GALA_LGE 1 1.35 187.7
4133 GALA_REG 1 1.45 145.2
4139 GRANNY_SMITH_REG 1 1.39 198.2
4017 GRANNY_SMITH_LGE 1 1.49 176.5
3115 PEACHES 1 2.09 145.5
4011 BANANAS 1 0.49 123.2
4383 MINNEOLAS 1 0.79 187.3
3144 TANGERINES 1 1.19 135.5
4028 STRAWBERRIES_PINT 0 0.99 104
4252 STRAWBERRIES_HALF_CASE 0 3.99 53
4249 STRAWBERRIES_FULL_CASE 0 7.49 67
94011 ORGANIC_BANANAS 1 0.99 56.3
```

Then, program should repeatedly invoke customer check-out functionality until the store associate (user) decides to quit. As part of checkout functionality, prompt for PLU code, validate it, then the user to input weight for each product if it is sold by weight, or # of units if sold by unit. Compute the price of the item and keep up the subtotal.

Once all purchased products are rung for a customer, output the total purchase amount. If the total purchase exceeds $50, apply 5% discount to the total.

We need to keep track of inventories automatically as well. So, keep updating the inventory data along with checkout operations. When the store closes every day, product information file should be saved in output.txt in the same format as input inventory file.

Implement this assignment in 3 files: product.h, product.cpp and store.cpp. When you are ready to submit your source files, zip them and submit as one package.

Assignment #2: Activities in a typical restaurant without using templates

This assignments gets into dynamic memory allocation big time! Unlike all previous assignments, this is a big one. That is why it is not due for a few weeks. I hope you will start on right away and enjoy the experience!

This assignment mimics the configuration and activities happen at a typical restaurant. Input is provided in 2 files: config.txt and activity.txt. Configuration file contains how many tables, table - waiter relationship & full menu list. Activity file mimics the actual activities that happen in restaurant. After setting up the necessary objects using the configuration file, you can read the activity file and process them. You should not use vector or any other template in this assignment.

We will use the following classes to complete this assignment. Instructor will provide the start-up files. Feel free to add more variables if needed. Avoid making drastic

changes to existing variables. You need to implement all the .cpp files including class implementation and overall application functionality.
Table : status, # of max seats, # of guests, order, waiter
menu_item: item_code, name, price
Menu : list of menu items
Order : a list of menu items ordered at a table
Sample configuration file (config.txt)

```
Tables: table #, max seats
1 2
2 4
3 2
4 2
5 2
6 4
7 6
8 10
9 2
10 4
11 4
12 4
13 4
14 2
15 2
16 2
17 2
18 2
Waiters: first name followed by table list
John 1,2,5,9,11,15
Maria 3,4,6,7,17,18
Mike 8,10,12,13,14,26
Menu: listing of the full menu: item code, name, price
A1 Bruschetta 5.29
A2 Caprese_Flatbread 6.10
A3 Artichoke-Spinach_Dip 3.99
A4 Lasagna_Fritta 4.99
A5 Mozzarella_Fonduta 5.99
E1 Lasagna_Classico 6.99
E2 Capellini_Pomodoro 7.99
E3 Eggplant_Parmigiana 8.99
E4 Fettuccine_Alfredo 7.49
E5 Tour_of_Italy 14.99
D1 Tiramisu 2.99
D2 Zeppoli 2.49
D3 Dolcini 3.49
S1 Soda 1.99
S2 Bella_Limonata 0.99
S3 Berry_Acqua_Fresca 2.88
```

Sample activity file (comments are not part of the file):

```
T1 P2 // Party of 2 is assigned to Table 1
T2 P4
T4 P2
T1 O A1 A1 E1 E2 // Party at table 1 orders these items
T8 P10
T1 S // Food arrives at table 1
T3 P2
T1 C // T1 gets the check, pays and leaves & table is cleaned too.
T5 P2
T1 P1 // Party of 1 is assigned to table 1
...
```

In addition to processing these commands, when a table is closed, display the check as well. Include table #, # of customers in the party, name and price of each ordered item and the total. No need to worry about tax or tips.
Error checking:
- Do not allow orders from table with no party assigned to it.
- Do not allow assigning new party to a table when another party is already there.
- Do not allow assigning new party to a table for which waiter has not been assigned.
- Do not allow check-out from empty table or a table in which all items have not been served.
- Do not allow delivery of food to an empty table!
- Do not assign any party to table with no waiter assignment.
- Reject any request that does not match with sample formats.

Assignment #3: Respin grocery checkout using template class
We are going to re-spin Project 2 using our own lookup table and handle bit different input format.

1. Design and use a generic lookup table template class to speed up lookup operation. However, we can assume that lookup key is integer. We are mapping it to an user-specified class. Lookup key should support multiple ranges. You can assume a MAX limit of 10 ranges. Our usage will be specifically for PLU codes which means ranges are 0000 to 9999 and 90000-99999, and we will be mapping it to Product class pointer. Skeletion code & data file are available under ~veerasam/students/proj4 directory.
2. Take full advantage of Tokenizer code we introduced during Project 2 to read new format of input file inventory.csv. I have attached it for your reference. Test.cpp contains sample usage - do not include it in your project.
3. Output should be generated in a new file called output.csv in the same format. We can use brute-force approach and use loops to check the whole ranges. Optionally, if you want additional challenge, you are welcome to review http://www.cs.northwestern.edu/~riesbeck/programming/c++/stl-iterator-define.html and write your own iterator class for the lookup table template class. Then, you can iterate through the contents and generate output.

Assignment #4: Respin restaurant using templates
Re-spin Assignment 2 using C++ template classes (primarily vector). Specifically, each array can be replaced by a vector. That means there is no need to assume max # of tables, max # of servers, etc. You can start with your assignment #2 solution and replace each array with a vector. All other requirements remain the same as Assignment #2.
Programming assignments will be graded on a 100 point basis, utilizing the following criteria:

		Max Score
Source Code	Overall design	40%
	Comments	10%
	Indentation	10%
	Readability	10%
Execution	Nominal cases	25%
	Special cases	5%
Total		**100%**

Policies and Procedures for Students
The University of Texas at Dallas provides a number of policies and procedures designed to provide students with a safe and supportive learning environment. Brief summaries of the policies and procedures are provided for you at http://provost.utdallas.edu/home/index.php/syllabus-policies-and-procedures-text and include information about technical support, field trip policies, off-campus activities, student conduct and discipline, academic integrity, copyright infringement, email use, withdrawal from class, student grievance procedures, incomplete grades, access to Disability Services, and religious holy days. You may also seek further information at these websites:

- http://www.utdallas.edu/BusinessAffairs/Travel_Risk_Activities.htm
- http://www.utdallas.edu/judicialaffairs/UTDJudicialAffairs-HOPV.html
- http://www.utsystem.edu/ogc/intellectualproperty/copypol2.htm
- http://www.utdallas.edu/disability/documentation/index.html

These descriptions and timelines are subject to change at the discretion of the Professor.