**COURSE SYLLABUS**

**2nd Semester, AY 2015-2016**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
| **Course Information** | | | | **Faculty Information** | | | | |
| **Course Code:** | | CpE 324N | | **Name:** | | | Luke Nigel Laylo | |
|  | | Data Structures and Algorithm | | **Office:** | | | Computer Engineering Department | |
| **Credit Units:** | | 3 units lecture; 1 unit laboratory | | **Email:** | | | *lukelaylo@gmail.com* | |
| **Pre-requisites:** | | CpE 311N (Introduction to Programming) | | **Phone:** | | | (032) 230 – 0100 (loc. 263) | |
| **Schedule:** | | 07:30 AM-10:30 AM MW Rm:  SELab  01:30 PM-04:30 PM MW Rm:  SELab | | **Consultation Time:** | | | 10:30 AM – 1:30 PM MTW | |
|  | | | | | | | | |
| **Course Description** | | | | | | | | |
| This course is intended for third-year undergraduate students of computer engineering. It essentially deals with the study of designing and implementing static and dynamic data structures and algorithms for systems and applications programming. The topics covered include algorithm implementation for recursive and iterative operations involving numerous data structure implementations that include its creation, destruction, insertion, deletion, traversals, sorting and searching. | | | | | | | | |
|  | | | | | | | | |
| **COURSE OUTCOMES (CO)** | | | | | | | | |
| *By the end of the semester, students should be able to:*  **CO1:** Synthesize data structures and algorithm appropriate for solving a given scenario;  **CO2:** Implement data structure and algorithm for solving a given scenario; | | | | **Alignment to Program Outcomes** | | | | |
| *The learning outcomes in this course are* ***enabling*** *to the achievement of:*  **[PO3**] Ability to design a system to meet desired needs in accordance with standards.  **[PO5]** Ability to identify, formulate, and solve computer engineering problems. | | | | |
|  | | | | | | | | |
| **ASSESSMENT OF OUTCOMES** | | | | | | | | |
| As evidence of having achieved the outcomes, students must produce quality outputs and/or carry out tasks successfully. | | | | | | | | |
| **Formative**  For **CO1**: Written Exercises (Non rubric – based)  For **CO2**: Hands – On Exercises  Practical Exercises | | | | **Summative (Rubric Based)**  For **CO1 and CO2**: Midterm Practical Exam  Final Practical Exams | | | | |
| **Assessment Rubrics (See Annex 1)**   |  |  | | --- | --- | | Output/Performance | Rubric No. & Title | | Hands – On Exercises | [CpE324N-100] Programming Exercise Rubrics | | Practical Exercises | [CpE324N-100] Programming Exercise Rubrics | | Practical Exam | [CpE324N-200] Programming Exam Rubrics | | | | | | | | | |
|  | | | | | | | | |
| **Grading System**  The student’s grade for the course is computed based on both formative and summative assessment data. The computation is detailed below. | | | | | | | | |
| **Grade Component Weight**  Midterm Practical Exam (20%)  Final Practical Exam (40%)  Hands – On Exercises (10%)  Practical Exercises (20%)  Class Standing (SQA) (10%) | | | | **Computation**  Midterm Practical Exam Grade x 0.20  Final Practical Exam Grade x 0.40  Hands – On Exercises Grade x 0.10  Practical Exercises Grade x 0.20  Class Standing Grade x 0.10  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Total **Grade** | | | | |
| **Passing Grade:** **3.0**  **Condition for Passing:** Rates from all grade components must be 3.0 or better. A compilation of hands-on exercises and practical exercises must be submitted at the end of the semester. | | | | | | | | |
|  | | | | | | | | |
| **LEARNING PLAN** | | | | | | | | |
| **Course Outcome** | **Topics** | | | | **Week** | **Learning Activities** | | |
|  | Introduction to the Course | | | | 1 | Open Forum | | |
| CO0 | **Unit 1: Review of Programming Basics**  Topics:   * Memory allocation and de-allocation using pointers. * Functions * System defined and user-defined data types, abstract data types. * Structures and pointers | | | | 2,3 | * Peer Teaching Exercises * Reading Assignments * Lab Exercises (individual): Abstract Data Type Application (statically allocated) implementing pointers, structures, functions * Seatwork: Programming review on pointers, structures, functions | | |
| CO1  CO2 | **Unit 2: Fundamental Abstract Data Structures (ADTs) Part 1**  Topics   * List ADT * Simple Array Implementation of Lists * Linked Lists * Doubly Linked Lists * Circularly Linked Lists * Cursor Implementation of Linked Lists | | | | 4-7 | * Peer-Teaching ExercisesLab Exercises (individual): Abstract Data Type Application (dynamically allocated) * Seatwork: List operations with comparison of static and dynamic implementation | | |
| CO1  CO2 | **Unit 3: Search, Sorting and Analysis of Algorithms**  Topics:   * Searching Algorithms * Sorting Algorithms * Algorithm Analysis | | | | 8,9 | * Peer-Teaching ExercisesWritten Exercise: sorting and searching number sets * Lab Exercises (individual): Abstract Data Type Application (dynamically allocated) applying searching and sorting algorithms * Seatwork: Searching and sorting random numbers | | |
|  | **Midterm Examination** | | | | 10 |  | | |
| CO1  CO2 | **Unit 4: Recursion**  Topics:   * Non – recursive implementation * Recursive implementation | | | | 11 | * Peer-Teaching Exercises * Programming Exercises: converting iterative function to recursive function * Lab Exercises (individual): Round Robin Scheduling | | |
| CO1  CO2 | **Unit 5: Tree ADTs**  Topics:   * General tree * Binary tree (expression tree) * Search tree ADT (binary search tree) * AVL tree (single & double rotation) * Splay tree * Heap tree | | | | 12-14 | * Peer-Teaching Exercises * Written Exercises: step-by-step output result of the basic operations * Lab Exercises (group): BST and Heap | | |
| CO1  CO2 | **Unit 6: Other ADTs**  Topics:   * Tries * Priority Queues (Heaps) * Hashing * Graphs * B-Trees | | | | 15-17 | * Peer-Teaching Exercises * Written Exercises: step-by-step output result of the basic operations * Lab Exercises (group): Hash Tables | | |
|  | **Final Examination** | | | | 18 |  | | |
|  | | | | | | | | |
| **Learning Resources** | | | | | | | | |
| **References**   * Van Wyk, Christopher J. Data structures and C programs. Reading, Mass. : Addison-Wesley (1988) * Weiss, Mark Allen. Data structures and algorithm analysis in C. 2nd ed.. Menlo Park, Calif. : Addison-Wesley (1997) * Wirth, Niklaus. Algorithms and data structure. London: Prentice-Hall ( 1986) * Goodrich, Michael T. Algorithm design: foundations, analysis, and Internet examples. New York : John Wiley (2002) * Aho, Alfred V. Data structures and algorithms. Reading, Mass : Addison-Wesley (1983) * Rowe, Glenn. An introduction to data structures and algorithms with Java /. -London; New York : Prentice Hall (1997) * Goodrich, Michael T., Tamassia, Robert. Data structures and algorithms in Java. 3rd ed. Hoboken, N.J. : J. Wiley (2004) * Amsbury, Wayne. Data structures: from arrays to priority queues. . -Belmont, California. Wadsworth Pub. * Drozdek, Adam; Simon, Donald L. Data structures in C. -Boston : PWS Pub.(1995) * Heileman, Gregory L. Data structures, algorithms, and object-oriented programming /. -New York : McGraw-Hill (1996) * Helman, Paul; Veroff, Robert; Carrano, Frank M. Intermediate problem solving and data structures: walls and mirrors /. - 2nd ed. -Redwood City, California. : Benjamin/Cummings Pub., (1991) * Kruse, Robert L.; Leung, Bruce P.; Tondo, Clovis L. Data structures and program design in C /. -Englewood California, N.J. : Prentice Hall (1991) * Naps, Thomas L.; Nance, Douglas W.; Singh, Bhagat. Introduction to computer science: programming, problem solving, and data structures /. - Alternate edition. -St. Paul, Minn. : West (1989) * Standish, Thomas A. Data structure techniques /. -Reading, Mass. : Addison-Wesley (1980) * Ken, Arnold, et. al. The Java Programming Language. 3rd ed. Reading, Massachusetts: Addison-Wesley (2000). * Data structures : a pseudocode approach with C by Gilberg, Richard F., ST Book 005.133 G37 c.1 * data structures techniques by standish,t., ST Book 001.642 st 24 * Data structures and C programs by Van Wyk, Christopher J., ST Book 005.73 V26 005.73 V26 * Data Structures and Software Development by Tremblay,Jean-Paul., ST Book * Data structures and the standard template library by Collins, William J., ST Book 005.73 C69 * Data structures and algorithms for game developers by Sherrod, Allen., ST Book 794.81526 Sh57, ST CD 794.81526 Sh57 c.1 * Data structures and algorithm analysis in C by Weiss, Mark Allen., ST Book 005.73 W43 * C & data structures by Deshpande, P. S., ENGG CD 005.133 D45, ST Book 005.133 D45 * Algorithms and data structure by Wirth, Niklaus., ST Book 005.73 W74 005.73 W74 * Introduction to computer science: programming, problem solving, and data structures. by Naps, Thomas L., ST Book 005 N16 1 * Data structures and software development in an object-oriented domain by Tremblay, Jean-Paul, SCI CD 005.117 T72 c.2, ST Book 005.117 T72 * Data structures, algorithms, and object-oriented programming by Heileman, Gregory L., ST Book 005.73 H36 * Reusable data structures for C by Sessions, Roger., ST Book 005.133 Se72 * Intermediate problem solving and data structures: walls and mirrors by Helman, Paul., ST Book 005.1 H36 005.1 H36 * Algorithm design by Kleinberg, Jon., ST Book 005.1 K67 * Fundamentals of computing II : abstraction, data structures, and large software systems, ST Book 001.64044 F96 1 * Algorithms in C by Sedgewick, Robert, ST Book 005.133 Se28 v.1 * Practical algorithms for programmers by Binstock, Andrew., ST Book 005.133 B51 * Introduction to algorithms by Cormen, Thomas H., ST Book 005.1 C81 * C : how to program by Deitel, Paul J., ST Book 005.133 D36   **Websites**   * Data Structures and Algorithms Course Notes, PLDS210 University of Western Australia. Retrieved July 06, 2005 from <.ciips.ee.uwa.edu.au/~morris/Year2/PLDS210/ds\_ToC.html> * Data Structures (Lecture Notes). These lecture notes are designed for on-line reference and review. Retrieved July 06, 2005 from <www.cs.sunysb.edu/~skiena/214/lectures/>   **Online Resources**  Use of online resources (e-books, tutorials, presentations, videos, lectures, and other supplementary materials) regarding further discussions on Data Structures and Algorithms are encouraged. | | | | | | | | |
|  | | | | | | | | |
| **Course Policies** | | | | | | | | |
| * **Attendance and Tardiness**   You are expected to attend all classes. The USC Student Manual (2013 Edition) stipulates that “a student who incurs unexcused absences of more than 20% of the prescribed number of class hours or laboratory periods during the term should be given NC or 5.0.” A 3-unit course has 48 class hours and a 1-unit laboratory course has 16 laboratory periods. You do the math.  Tardiness is highly discouraged and habitual tardiness will not be condoned. Appropriate sanctions for tardiness will be given based on agreement reached during a one-on-one conference between you and me. If you come late to class, silently make your way to your seat without disrupting ongoing activity and approach me at the end of the class to have your attendance checked.   * **Use of Gadgets in Class**   Gadgets should only be used in class in aid of learning. It’s allowable that you go online in the classroom if you want to find out more about something on the topic being taken up. In no way that you are allowed to use your gadgets in class to do social networking, games, or other activities that have no direct bearing on the ongoing class activity. You may take pictures of what is written on the board but only after I expressly announce when you can do it. At all times, set your gadgets on silent mode.   * **Problem Sets**   Students are encouraged to work together on the problem sets and when studying. However, the solutions that you shall turn in must be your own work, not a direct copy of someone else’s work. Cases of copying shall be dealt with following the university’s procedures for disciplinary actions. Note that the university considers dishonesty or any fraudulent act as a major offense. Thus, make sure that you do your own work and that you protect them from plagiarism by others.  Solutions to problem sets that are submitted late will not be accepted.   * **Missed Tests and Exams**   If you miss to take a test/examination, you can make a formal written request to take a special test/examination in cases when you missed a test/examination due to serious medical condition or emergency reasons. "EMERGENCY shall be understood as an unforeseen combination of circumstances which calls for an immediate response to an urgent need for assistance or relief.” Pertinent supporting documents must be attached to your letter of request.   * **Examinations**   Note that there are six (6) exams that you must take during the semester. In case you missed an examination due to serious medical condition or emergency reasons, you must make a formal written request to take a special examination. "EMERGENCY shall be understood as an unforeseen combination of circumstances which calls for an immediate response to an urgent need for assistance or relief.” Pertinent supporting documents must be attached to your letter of request. Otherwise, a grade of 5.0 will be recorded on that missed examination.   * **Consultation**   My consultation periods are indicated in this syllabus. Should you wish to consult with me on matters pertaining to your achievement of the learning outcomes, you can inform me through email at least 24 hours before. Please indicate in clear terms what you wish to consult with me. You may do so individually or as a team/group.   * **Communication**   All course-related communications outside of class should be done through email. I will send to you supplementary learning materials, announcements, instructions, and the like through this method so make sure you regularly check your inbox. | | | | | | | | |
|  | | | | | | | | |
| Prepared by | | |  | Approved by | | | |  |
| **Luke Nigel Laylo**  Faculty | **Engr. Antoniette P. Mondigo** Department Chair |
| Date Submitted for Approval | | | **Nov. 21, 2015** | Date Approved | | | |  |

**ANNEX 1**

**CpE 324N – Data Structures and Algorithm**

**A complete list of grading rubrics for CpE 324N**

**[CpE324N-100] Programming Exercise Rubrics**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Criteria** | **Weight** | **Expert(1.0)** | **Proficient (1.3)** | **Competent(2.0)** | **Novice (3.0)** | **Beginner (5.0)** |
| **Specification based Analysis** | 50% | Includes modules/features whose usefulness/purpose are well justified. | Meets 100% of the problem specification functionalities. | Meets 80% of the problem specification functionalities. | Meets 60% of the problem specification functionalities. | Lesser than 60% of the problem specification functionalities. |
| **Code Efficiency** | 35% | Code is extremely efficient both memory and execution without sacrificing readability and understanding. | Optimizes the code either the memory or execution without sacrificing readability and understanding. | The code is fairly efficient without sacrificing readability and understanding. | The code is brute force and unnecessary long. | The code is huge and appears to be patched together. |
| **Robustness** | 5% | Program handles erroneous or unexpected failures gracefully and actions follow certain industry standards. | 100% of the error conditions are handled correctly. | 80% of the test cases are handled correctly. | 60% of the test cases are handled correctly.  Contains warnings. | Less than 60% of the test cases are handled correctly. |
| **Readability & Modularity** | 5% | Concise code is used. Documents code fragments that represent uncommon process and complex algorithm. | Code is concise, well – organized and easy to maintain. The coding convention introduced in class is followed 100% of the time. Places few comments to describe code snippets. | Code is partially organized. 80% of the identifiers are meaningful and descriptive. Minor inconsistencies in indention. Does not place comments to describe code snippets. | Code is less organized and not easy to maintain and trace. 60% of the identifiers are not meaningful and descriptive. Few inconsistences in indention. Doesn’t place comments to describe code snippets. | Does not meet the 60% capability of creating code organization. |
| **Delivery** | 5% | Program was delivered before the set time. | Program was delivered on time. | Program was delivered on the week. | Program was delivered on the week after. | Program was delivered for the sake of compilation. |

**[CpE324N-200] Programming Exam Rubrics**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Criteria** | **Weight** | **Expert(1.0)** | **Proficient (1.3)** | **Competent(2.0)** | **Novice (3.0)** | **Beginner (5.0)** |
| **Specification based Analysis** | 50% | Includes modules/features whose usefulness/purpose are well justified. | Meets 100% of the problem specification functionalities. | Meets 80% of the problem specification functionalities. | Meets 60% of the problem specification functionalities. | Lesser than 60% of the problem specification functionalities. |
| **Code Efficiency** | 30% | Code is extremely efficient both memory and execution without sacrificing readability and understanding. | Optimizes the code either the memory or execution without sacrificing readability and understanding. | The code is fairly efficient without sacrificing readability and understanding. | The code is brute force and unnecessary long. | The code is huge and appears to be patched together. |
| **Robustness** | 5% | Program handles erroneous or unexpected failures gracefully and actions follow certain industry standards. | 100% of the error conditions are handled correctly. | 80% of the test cases are handled correctly. | 60% of the test cases are handled correctly.  Contains warnings. | Less than 60% of the test cases are handled correctly. |
| **Readability & Modularity** | 5% | Concise code is used. Documents code fragments that represent uncommon process and complex algorithm. | Code is concise, well – organized and easy to maintain. The coding convention introduced in class is followed 100% of the time. Places few comments to describe code snippets. | Code is partially organized. 80% of the identifiers are meaningful and descriptive. Minor inconsistencies in indention. Does not place comments to describe code snippets. | Code is less organized and not easy to maintain and trace. 60% of the identifiers are not meaningful and descriptive. Few inconsistences in indention. Doesn’t place comments to describe code snippets. | Does not meet the 60% capability of creating code organization. |

**[CpE324N-300] Collection of Written Rubrics**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Criteria** | **Weight** | **Expert(1.0)** | **Proficient (1.3)** | **Competent(2.0)** | **Novice (3.0)** | **Beginner (5.0)** |
| **Analysis** | 60% | Makes a full comparison (more than two) that has a complete evidence to back-up the claim. | Satisfy the specification but with few minor omissions. | Understanding of the concept is visible. | Significantly incomplete. | Irrelevant analysis. |
| **Clarity and Completeness** | 40% | All parts has been shown. | 80% has been shown correctly. | 50% has been shown correctly. | Started correctly but not able to continue further process. | No answer given. Wrong start. |