|  |  |
| --- | --- |
| **ELEMENT** | **CONTENT** |
| DEPARTMENT | CIS |
| AUTHOR (S) | Peter Chapin, Leslie Damon |
| COURSE NUMBER | **CIS 3050** |
| COURSE TITLE | **Algorithms & Data Structures** |
| SHORT TITLE | Algorithms/Data Struct |
| COURSE LEVEL | 3000 |
| DATE CREATED |  |
| CHECKED/CHANGED | 6/7/2017 |
| PREREQUISITES | CIS 2010; CIS 2025 or 2260 |
| COREQUISITES |  |
| RESTRICTIONS |  |
| SPECIAL FEES | No |
| CREDITS | 3 |
| HOURS | 3 hours of lecture per week |
| SEMESTER | Fall |
| COURSE DESCRIPTION | This course focuses primarily on the implementation of various important algorithms and data structures. It contains some theory, but the theory content is minimized in favor of a more rigorous treatment of implementation techniques. The course covers classic topics such as lists, trees, hash tables, graphs, sorting, and string matching. Other topics such as encryption, data compression, and image processing are covered as time allows. |
| SUGGESTED TEXTS | *Data Structures and Algorithms in C++*; Goodrich, Tamassia, & Mount |
| OPTIONAL TEXTS |  |
| COURSE OUTCOMES | The successful student will be able to:   1. Write a production-quality implementation of an algorithm or data structure using C++ following the style of the standard template library 2. Understand the operation of several sorting algorithms 3. Understand the operation of linked lists, binary search trees, and hash tables 4. Understand the operation of queues, stacks, and heaps 5. Understand the operation of at least one graph algorithm 6. Understand the concepts of exception safe coding and provide exception safety if necessary 7. Write a professional-quality test program to exercise library code 8. Understand the concepts of asymptotic running time and recurrences with appropriate mathematical background 9. Read and understand a theoretical description of an algorithm or data structure along with its proofs of correctness and performance characteristics |
| COURSE CONTENT | 1. Review of C    1. Pointers    2. Malloc    3. Free 2. Sorting    1. Insertion sort    2. Merge sort 3. Asymptotic running time: recurrences    1. Sorting    2. Heap sort    3. Using heaps as priority queues    4. Quick sort 4. Linked lists 5. Binary search trees 6. Hash table    1. Chaining    2. Open addressing 7. Graph algorithms 8. Implementation topics    1. Exceptions    2. Testing 9. Selected topics as time allows |
| LAB/STUDIO OUTCOMES |  |
| LAB/STUDIO CONTENT |  |
| LECTURE CAPACITY | 32 |
| LAB CAPACITY |  |
| GRADED OR P/NP | Graded |
| EVALUATION |  |
| DELIVERY METHOD | LEC |
| ROOM REQUIREMENTS |  |
| AUTHOR’S NOTES |  |