# Undergraduate Course Syllabus

<table>
<thead>
<tr>
<th>Course ID</th>
<th>311037030</th>
<th>Course Name</th>
<th>Practice in Data Structure &amp; Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>■ Compulsory □ Selective</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Course</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>■ English □ Chinese</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Credit Hour</td>
<td>3</td>
<td>Period</td>
<td>48</td>
</tr>
<tr>
<td>Semester</td>
<td>□ First Fall □ First Spring □ Second Fall ■ Second Spring □ Third Fall □ Third Spring □ Fourth Fall □ Fourth Spring</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instructors</td>
<td>Li xiaohua Zuo hang</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>This course thoroughly covers key data structures at the undergraduate level. With a focus on how to assess costs and benefits, it teaches students how to create efficient data structures and algorithms and how to adopt to new design challenges. Students are taught how to assess applications needs to find data structures with matching capabilities.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prerequisites</td>
<td>C Language Programming Discrete Mathematics Data Structure &amp; Algorithmic</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3) Clifford A. Shaffer[] A Practical Introduction to Data Structures and Algorithm Analysis——— C++[]) Publishing house of electronics industry. |
| Resource    | Class Participation (40%), Final Report (60%) |
Topics

Tools & Environment

This course will require to use VC++ 6.0 software for programming

Experiment 1 6h
Implement a class of a singly linked list of integers, named IntLinkedList. Define the following methods in the class.

a. Void InsertFirst(int); // insert the value in parameter at the first position in the current linked list
b. Void InsertLast(int); // insert the value in parameter at the last position in the current linked list
c. Int DeletetFromFirst(); // delete the first node and return its value.
d. Int DeleteFromLast(); // delete the last node and return its value.
e. Void DeleteNode(int); // delete nodes whose value is given in the parameter.
f. Bool IsInList(int); // check whether the value in parameter is in linked list.
g. Int Length(); // return the number of nodes in the linked list.
h. IntLinkedList* Clone(); // return a pointer to a new copy of current linked list.
i. ~IntLinkedList(); // delete all nodes in the linked list.

Experiment 2 4h
Merge two ordered singly linked lists of integers into one ordered list.
L1 = ( 1 4 5 8 12 49) L2 = (0 17 22 45 100)

Experiment 3 4h
Implement a class of doubly linked list of integers, named IntDoublyLinkedList. Define the following methods in the class.

a. IntDoublyLinkedList(IntLinkedList*); // Create a doubly linked list from a singly linked list.
b. Void InsertFirst(int); // insert the value in parameter at the first position in the current linked list
c. Void InsertLast(int); // insert the value in parameter at the last position in the current linked list
d. Void Delete(int); // delete all nodes whose value is given in the parameter.
e. Void InsertMiddle(int); // insert a node in the middle of a doubly linked list.
<table>
<thead>
<tr>
<th>Experiment</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4h</td>
</tr>
<tr>
<td>5</td>
<td>4h</td>
</tr>
<tr>
<td>6</td>
<td>4h</td>
</tr>
<tr>
<td>7</td>
<td>4h</td>
</tr>
<tr>
<td>8</td>
<td>6h</td>
</tr>
<tr>
<td>9</td>
<td>6h</td>
</tr>
</tbody>
</table>

**Experiment 4** 4h
Implement a program to implement “Reverse a number series” Test your program with the number series 1, 3, 5, 7, 9, 2, 4, 6, 8.

**Experiment 5** 4h
Implement a program to implement “Convert decimal to binary”, Test it with the numbers 19, 127, and 255.

**Experiment 6** 4h
Implement the program for an algorithm called *stackToQueue* that creates a queue from a stack. After the queue has been created, the top of the stack should be the front of the queue and the base of the stack should be the rear of the queue. At the end of the algorithm, the stack should be empty.

**Experiment 7** 4h
Implement the program for an algorithm called *queueToStack* that creates a stack from a queue. After the stack has been created, the front of the queue should be the top of the stack and the rear of the queue should be the base of the stack. At the end of the algorithm, the queue should be empty.

**Experiment 8** 6h
Implement the program for an algorithm called *reverseStack* that reverses the contents of a stack. The algorithm must have only one parameter for the stack to be reversed. Hint: use a temporary queue.

**Experiment 9** 6h
Implement these exercises using C++
P1. Count the leaves of a binary tree at a given level
P2. Delete all leaves of a binary tree whose value equals a give number.
P3. Print a binary tree in breath-first search order.
Ex.
Result: A B F C G D E H I J K

Experiment 10   6h
Write a program that reads a list of names and telephone numbers from a text file and inserts them into a BST tree. Once the tree has been built, present the user with a menu that allows him or her to search the list for a specified name, insert a new name, or print the entire phone list. At the end of the job, write the data in the list back to the file. Test your program with at least ten names.