1. **Using the Node struct:**
    1. Write a function that initializes a node. Initialize a node with the value {“123”, “Nguyen Van A”, 2004}
    2. Write a function that prints the value of a node.
    3. Print the value of the node just initialized.

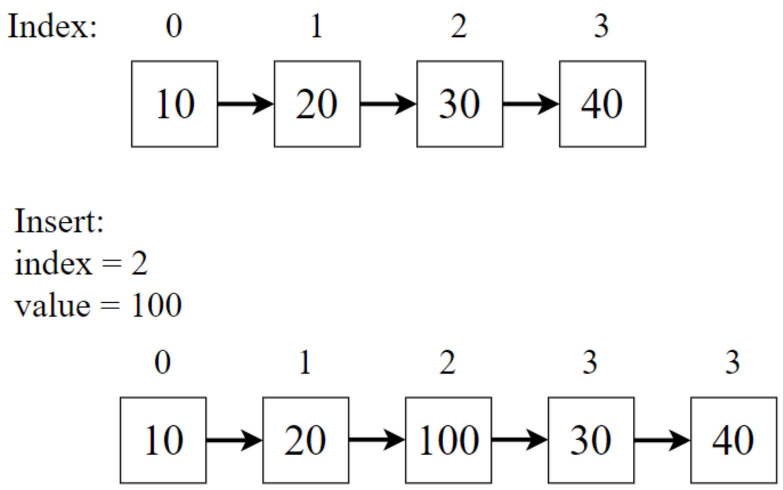
input	ouput
123 Nguyen Van A 2004	123 Nguyen Van A 2004

2. **Add node to linked list.**
    1. Write a function that adds a node to the end of the linked list. With the function's input parameter: Append_Node_2_Linked_List (string student_id, string name, int birth_year)
    2. Write a function that prints the value of the linked list.
    3. Create a linked list with value {“123”, “Nguyen Van A”, 2004}.
    4. Print the value of the linked list.

input	output
123 Nguyen Van A 2004	123 Nguyen Van A 2004

3. **Insert Value**

Write a function that inserts a value into the linked list at position i. Assume i is always less than the number of nodes. For example:



a. Write a function that inserts a node into the linked list at position i with the function's input parameter: Insert_Node_2_Linked_List (string student_id, string name, int birth_year, int index)
b. Create a linked list with the values {“123”, “Nguyen Van A”, 2004}, {“125”, “Vo Van C”, 2004}. For simplicity, students can initialize without typing std in.
c. Insert value {“124”, “Tran Thi B”, 2004} in index = 1
d. Print out the value of the linked list.

input	output
124 Tran Thi B 2004 1	123 Nguyen Van A 2004 124 Tran Thi B 2004 125 Vo Van C 2004

4. Delete students.

1. Write a function to delete students in the linked list by student_id. Input parameter of function: Delete_Node_By_Id(string deleted_student_id)

2. Create a linked list with the values {“123”, “Nguyen Van A”, 2004}, {“124”, “Tran Thi B”, 2004}, {“125”, “Vo Van C”, 2004}. For simplicity, students can initialize without typing std in.
3. Delete student with student_id = 125.
4. Print out the value of the linked list.

input	**output**
125	123 Nguyen Van A 2004 124 Tran Thi B 2004

5. Rename of student:

1. Write a function that allows you to edit the student's name (name) based on student_id. Input parameter of function: Modify_Name_By_Id(string student_id, string new_name)
2. Create a linked list with the values {“123”, “Nguyen Van A”, 2004}, {“124”, “Tran Thi B”, 2004}, {“125”, “Vo Van C”, 2004}. For simplicity, students can initialize without typing std in.
3. Change the name of the student with student_id = 125 to "Truong Thi Z".
4. Print out the value of the linked list.

input	**output**
125 Truong Thi Z	123 Nguyen Van A 2004 124 Tran Thi B 2004 125 Truong Thi Z 2004

6. **Adding node to tree:**
    1. Write a function that adds a node to the AVL tree.
    2. Add nodes: 10, 100, 20, 30, 40, 50 in turn to the AVL tree. For simplicity, students can initialize without entering std in.
    3. Write a function that prints the value of the tree, in order of inorder (left - root - right)
    4. Print the value of the entire AVL tree.

input	**output**
	10 0 30 40 50 100

7. **Remove node from tree:**
    1. Write a function to remove a node from the AVL tree.
    2. Add nodes: 10, 100, 20, 30, 40, 50 to the AVL tree. For simplicity, students can initialize without entering std in.
    3. Enter the key of the value to be deleted, with the value key = 10.
    4. Write a function that prints the value of the tree, in order of inorder (left - root - right).
    5. Print the value of the entire AVL tree.

input	**output**
10	20 0 40 50 100

8. **Check the node in the tree:**
    1. Write a function to check whether the AVL tree contains a node with the value key = x or not? If yes, print “exist”, otherwise print “non-exist”.
    2. Add nodes: 10, 100, 20, 30, 40, 50 in turn to the AVL tree. For simplicity, students can initialize without entering std in.
    3. Find if node=50 exists in AVL tree?

input	**output**
50	exist

9. **Sum of two nodes in the tree:**
    1. Write a function to check, does the AVL tree exist two nodes, such that the sum of those two nodes is equal to the given sum? If yes, print to the screen “exist”, otherwise print “non-exist”.
    2. Add nodes: 10, 100, 20, 30, 40, 50 to the AVL tree. For simplicity, students can initialize without entering std in.
    3. Check the AVL tree with two nodes whose sum is 80 or not?

input	**output**
80	exist

10. **The kth largest value:**
    1. Write a function to find the kth largest value in the AVL tree. For example: [10, 20, 30, 40], the 2nd largest value is 30.
    2. Add nodes: 10, 100, 20, 30, 40, 50 to the AVL tree. For simplicity, students can initialize without entering std in.
    3. Find the k = 2 largest values in the AVL tree and print it to the screen.

input	**output**
2	50