# Online Assignment 1: List (30/11/2021 Tuesday-Morning)

You need to write new List implementations (both Arr based and LL based) that have all the functions of the List ADT as per the specification of your offline assignment (i.e., Table 1 of the relevant document) plus the following functions (c.f. Table 1A). You need to use your Arr based and LL based implementations to implement the functions defined in Table 1A to write your new implementations without changing any existing methods.

Table 1A: Extended functions

| Fn # | Function | Param. | Ret. | After functions execution | Comment |
|---|---|---|---|---|---|
| | [before function execution] | | | <20, 23 \| 12, 15, 1> | The vertical bar indicates the current position. Here the current position is 2 (first position is 0) and corresponds to the element 12. |
| 14 | update(item) | 19 | | <20, 23 \| 19, 15, 1> | update the element at the current location. |
| 15 | duplicate() | | | <20, 23 \| 12, 12, 15, 1> | Current element is duplicated. |

**Clerk Input Simulation (CIS)**

Now create a list of student information where each element contains a tuple <x, y>, where x is a student no. and y is a course number and <x, y> means that student x has registered for the course y (for simplicity of taking inputs, assume that for x and y are integers). Now, a clerk is inputting the information from a collection of tuples and she follows the following protocol for an input tuple <x, y>: if the current position also contains information of either x or y, she **duplicates** that and then **updates** the inserted tuple appropriately; otherwise, she **inserts** the tuple. Now you need to simulate this inputting process.

Input: you will have a number of lines containing a couple of integers, space separated, representing the tuple <x,y>. The end of input is marked by two 0s.

Output: you will have to output the ‘action’ of the clerk and the state of the underlying list due to that action in the following line.

Example Input:

```
1 101

2 102

3 102

3 103

0 0
```

**Example Output:**

```
Insert

[1 101]

Insert

[1 101] [2 102]

duplicate

[1 101] [2 102] [2 102]

update

[1 101] [2 102] [3 102]
```

duplicate

[1 101] [2 102] [3 102] [3 102]

update

[1 101] [2 102] [3 102] [3 103]

**Submission Guidelines:**

1. **Create a directory with your 7-digit student id as its name**
2. **You need to create separate files for the Arr implementation code (e.g. Arr1.cpp/Arr1.py), LL implementation code (e.g. LL1.cpp/LL1.py) putting common codes in another file. You will also include your previous implementations as separate files. Create a separate file for the main function implementing the CIS (e.g., CIS.cpp/CIS.py). No built in data structures can be used.**
3. **Put all the source files only into the directory created in step 1. Also create a readme.txt file briefly explaining the main purpose of the source files.**
4. **Zip the directory (compress in .zip format. Any other format like .rar, .7z etc. is not acceptable)**
5. **Upload the .zip file on Moodle in the designated assignment submission link. For example, if your student id is 1905xxx, create a directory named 1905xxx. Put only your source files (.c, .cpp, .java, .h, etc.) into 1905xxx. Compress the directory 1905xxx into 1905xxx.zip and upload the 1905xxx.zip on Moodle.**

***Failure to follow the above-mentioned submission guideline may result in upto 10% penalty.***