tcg crest
Inventing Harmonious Future

**Institute for Advancing Intelligence, TCG CREST**
(TCG Centres for Research and Education in Science and Technology)

# Introduction to Computer Programming and Data Structures
# Assignment 06

Maximum Marks: **200** Submission Deadline: **2022-Sep-30**
Bonus: **20** – programming style and efficiency
Topic: Lists and Sets

**Assignment problem # AP0601**

**Complex number system**: Write a structure COMPLEX having two components– x and y as `typedef struct complex { float x; float y; }COMPLEX;`.
Thus a structure COMPLEX z=x+iy will have a real value x and an imaginary value y. Now, do the following operations systematically one-by-one.

1. $z \leftarrow complex\_init(x, y)$: takes a real-imaginary pairs of floating numbers, allocates memory for a complex structure and initiate its values with the received values.

2. $z_3 \leftarrow complex\_add(z_1, z_2)$: Takes two COMPLEX structures and a return a new complex structure that holds the sum of the given complex numbers. Thus $z_3 = z_1 + z_2$.

3. $z_3 \leftarrow complex\_sub(z_1, z_2)$: Takes two COMPLEX structures and a return a new complex structure that holds the subtraction of $z_2$ from $z_1$. Thus $z_3 = z_1 - z_2$.

4. $z_3 \leftarrow complex\_mult(z_1, z_2)$: Takes two COMPLEX structures and a return a new complex structure that holds the multiplication of the given complex numbers. Thus $z_3 = z_1 * z_2$.

5. $z_3 \leftarrow complex\_div(z_1, z_2)$: Takes two COMPLEX structures and a return a new complex structure that holds the division $z_3 = z_1/z_2$. Thus $z_3 = z_1/z_2$.

6. $b \leftarrow complex\_print(z)$: Takes a COMPLEX structure prints the element in the form "x + i y". Return a success bit $b$ in the end of successful printing.

7. Write your own main function to tests the above functions.

- For inputs and outputs, follow #AP0303

[20]

**Assignment problem # AP0602**

**Lists:** Create a linked list of complex numbers. Each node is structure as `typedef struct complex_node { COMPLEX z; struct complex_node * next; }COMPLEX_NODE;`. Now, do the followings.

1. $N \leftarrow Create\_Complex\_Node()$: That creates a node and returns the address of the node.

2. $b \leftarrow List\_Add(L, z)$: Take the address of a list $L$ and add an element $z$ in the end of the list $L$. It returns a bit that indicates success/failure. ($L$ can be considered as head of the list)

3. $b \leftarrow List\_Push(L, z)$: Take the address of a list $L$ and add an element $z$ in front of the list $L$. It returns a bit that indicates success/failure.

4. $b \leftarrow List\_Delete(L, z)$: Take the address of a list $L$ and delete the element $z$ if exists. It returns a bit that indicates success/failure.

5. $b \leftarrow Show\_Complex\_List(L)$: Take the address of a list $L$ and prints the elements present in the list. It returns a bit $b$ that indicates success/failure.

6. $b \leftarrow List\_Search(L, z)$: Take the address of a list $L$ and search the element $z$. It returns $b = 1$ if exists, $b = 0$ if does not exist and $b = -1$ if fails to search.

7. $b \leftarrow List\_Sort(L)$: Take the address of a list $L$ and sort the elements according to the distances from origin.

8. $b \leftarrow List\_Insert(L, z, pos\_n)$: Insert an complex number $z$ in the $pos\_n^{th}$ position in the list. It returns a bit that indicates success/failure.

9. $x \leftarrow List\_Pop(L)$: Take the address of a list $L$ and delete the element $z$ from the front of the linked list. It returns a bit that indicates success/failure.

10. $b \leftarrow List\_Reverse(L)$: Take the address of a list $L$, rearrange the values in reverse order. It returns a bit $b$ that indicates success/failure.

11. $L_3 \leftarrow List\_Concatenate(L_1, L_2)$: Takes two lists $L_1$ and $L_2$, create a new empty list $L_3$. Rearrange the lists so that $L_3$ contains the complex numbers from $L_1$ and $L_2$.

12. $l \leftarrow List\_Length(L)$: Returns the length of the linked list $L$.

13. $z \leftarrow List\_Max(L)$: Returns the complex number $z$ having maximum distance from the origin.

14. $z \leftarrow List\_Min(L)$: Returns the complex number $z$ having minimum distance from the origin.

15. $L_3 \leftarrow List\_union(L_1, L_2)$: It takes two linked lists $S_1$ and $S_2$ and returns a new list $S_3$ that contains all elements from $L_1$ and $L_2$ without repeat. Note that *concatenate* may return a list having nodes with same complex numbers. Even any list can have repeated complex number. Thus $L_3 \leftarrow L_1 \cup L_2$

16. $L_3 \leftarrow List\_intersection(L_1, L_2)$: It takes two linked lists $L_1$ and $L_2$ and returns a new list $L_3$ which contains those complex numbers which are contained in both given lists. Thus, $L_3 \leftarrow L_1 \cap L_2$

17. $L_3 \leftarrow List\_difference(L_1, L_2)$: It takes two linked lists $L_1$ and $L_2$ and returns a new list $L_3$ which contains those complex numbers which are contained in $L_1$ but not in $L_2$. $L_3 \leftarrow L_1 \setminus L_2$

- Write your own main function having a meaningful menu to test the above operations.

[200]