Experiment set 1

Structure and Linked List

Structures :

The format for defining a structure is

```
struct Tag {
  Members
};
```

Where Tag is the name of the entire type of structure and Members are the variables within the struct.

To actually create a single structure the syntax is

```
struct Tag name_of_single_structure;
```

To access a variable of the structure it goes

```
Name_of_single_structure.name_of_variable;
```

For example if you want to create structure for storing a complex number, you will have to write

```
struct ComplexNumber {
  float real_part;
  float imaginary_part;
};
```

To create a new complex number structure and initialize it to 2 + 3i you will write

```
struct ComplexNumber z;
z.real_part = 2;
z.imaginary_part = 3;
```

You can define pointers to structures in the same way as you define pointer to any other variable

```
struct tag *struct_pointer;
```

```
struct ComplexNumber *zptr;
```

Now, you can store the address of a structure variable in the above defined pointer variable. To find the address of a structure variable, place the '&'; operator before the structure's name as follows

```
struct_pointer = &Name_of_single_structure;
```

```
zptr=&z
```

To access the members of a structure using a pointer to that structure, you must use the → operator as follows −

```
struct_pointer->title;
```

```
printf("%f+i%f",zptr->real_part, zptr->imaginary_part)
```

Ex1 : Write a program in C which

1. **Create a structure tagged as 'student' which contains two members**
    a. **One string variable "name"**
    b. **One Integer variable "ID"**
2. **Declare and initialize a structure "stu1" with name and ID**
3. **Pass the structure "stu1" to a function named "PrintStruc" which will print element of structure.**
4. **Pass a pointer to structure "srtptr" to a function and change the name and ID as "abc" and '123' respectively print the new structure member values in the main function using structure pointer.**

Dynamic Memory Allocation :

Dynamic memory allocation allows your program to obtain more memory space while running, or to release it if it's not required.
There are 4 library functions under "`stdlib.h`" for dynamic memory allocation.

Function	Use of Function
malloc()	Allocates requested size of bytes and returns a pointer first byte of allocated space `ptr = (cast-type*) malloc(byte-size)`

	The `malloc()` function returns a pointer to an area of memory with size of byte size. If the space is insufficient, allocation fails and returns NULL pointer. `ptr = (int*) malloc(N * sizeof(int));`
calloc()	Allocates space for an array elements, initializes to zero and then returns a pointer to memory. The name calloc stands for "contiguous allocation". `ptr = (int*) calloc(N, sizeof(int));` This statement allocates contiguous space in memory for an array of 25 elements each of size of float, i.e, 4 bytes.
free()	deallocate the previously allocated space. `free(ptr)`
realloc()	Change the size of previously allocated space `ptr = (int*) realloc(ptr,N+2)`

Ex2: Write a program in C to find the largest student “name” with largest “ID” with dynamic memory allocation.

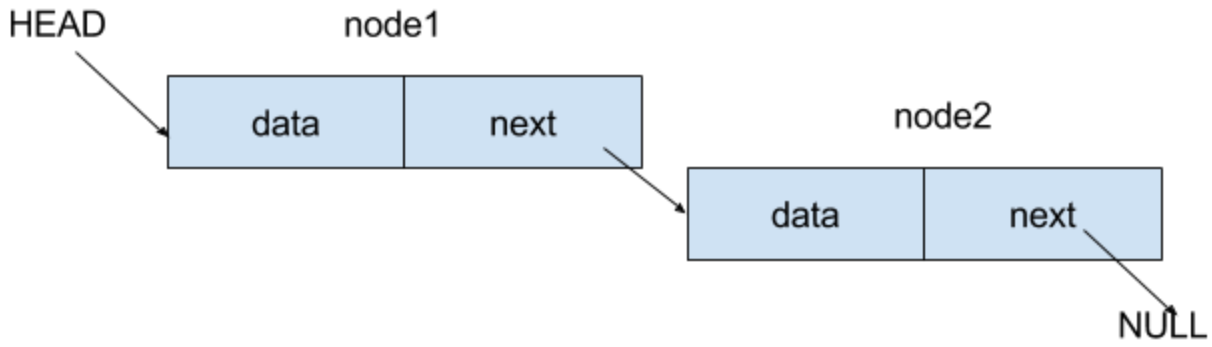
1. **Declare a structure pointer “list” [hint: struc student *list]**
2. **Take input a number “N”**
3. **Dynamically allocate memory to structure pointer list for N structure**
4. **Take input “name” and “ID” for “N” student**
5. **Print the name of student with largest ID.**

Linked List:

A linked list is a set of dynamically allocated nodes, arranged in such a way that each node contains one value and one pointer. The pointer always points to the next member of the list. If the pointer is NULL, then it is the last node in the list.

A linked list is held using a local pointer variable which points to the first item of the list. If that pointer is also NULL, then the list is considered to be empty.

```
/* Node of a linked list */
struct Node
{
  int data;
  struct Node *next; // Pointer to next node in DLL
};
```



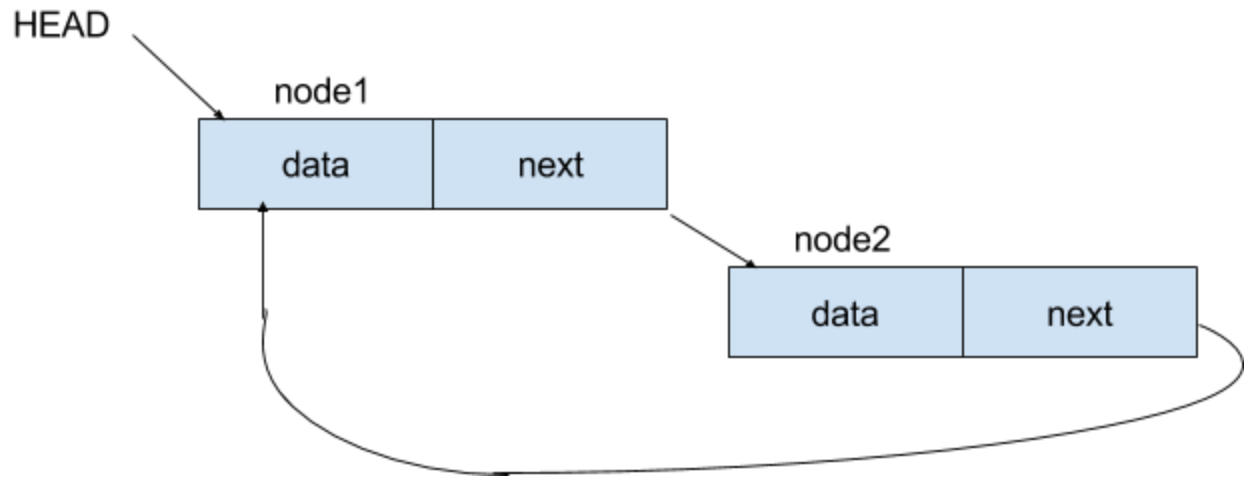
Ex3 : Write a program in C which

1. **Create a Linked List of "node" where "node" is a structure with following structure**
    a. **One int variable "data"**
    b. **One pointer to "node"**
2. **Add "N"(user given) nodes to linked list iteratively one at a time at the beginning of list.**
3. **Print all data in the list**
4. **Add a node after a node containing "data=val".**
5. **Delete a nodes from the above linked list which having "data=val"**
6. **Reverse the list and print values.**

Doubly Linked List:

```
/* Node of a doubly linked list */
struct Node
{
  int data;
  struct Node *next; // Pointer to next node in DLL
  struct Node *prev; // Pointer to previous node in DLL
};
```

Circular Linked List:



Ex4: Write a program in C to search an element in a circular linked

Homework :

Ex5 : Repeat Ex2 with doubly linked list
Ex6 : Build two circular linked list of different size such that number of node in the first list is more than number of node in second.Then merge a linked list into another circular linked list by inserting nodes of second list into first list at alternate positions of first list.