**Assignment Day 3**

**Question 1**

Write a function “insert\_any()” for inserting a node at any given position of the linked list. Assume position starts at 0.

**Answer:**

void insertPos(Node\*\* current, int pos, int data)

{

    // This condition to check whether the

    // position given is valid or not.

    if (pos < 1 || pos > size + 1)

        cout << "Invalid position!" << endl;

    else {

        // Keep looping until the pos is zero

        while (pos--) {

            if (pos == 0) {

                // adding Node at required position

                Node\* temp = getNode(data);

                // Making the new Node to point to

                // the old Node at the same position

                temp->next = \*current;

                // Changing the pointer of the Node previous

                // to the old Node to point to the new Node

                \*current = temp;

            }

            else

              // Assign double pointer variable to point to the

              // pointer pointing to the address of next Node

              current = &(\*current)->next;

        }

        size++;

    }

}

**Question 2**

Write a function “delete\_beg()” for deleting a node from the beginning of the linked list.

**Answer:**

Node\* removeFirstNode(struct Node\* head)

{

    if (head == NULL)

        return NULL;

    // Move the head pointer to the next node

    Node\* temp = head;

    head = head->next;

    delete temp;

    return head;

}

// Function to push node at head

void push(struct Node\*\* head\_ref, int new\_data)

{

    struct Node\* new\_node = new Node;

    new\_node->data = new\_data;

    new\_node->next = (\*head\_ref);

    (\*head\_ref) = new\_node;

}

**Question 3**

Write a function “delete\_end()” for deleting a node from the end of the linked list.

**Answer:**

//Function to delete a node from the end of a linked list.

void delete\_end()

{

if(header->link == NULL)

{

printf("\nEmpty Linked List. Deletion not possible.\n");

}

else

{

//Traverse to the end of the list.

ptr = header;

while(ptr->link != NULL)

{

ptr1 = ptr;

ptr = ptr->link;

}

ptr1->link = ptr->link;

free(ptr);

printf("\nNode deleted from the end.\n");

}

}