Assignment Day 3 | 26th December 2020

Question 1.

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

Sol:

```
#include <stdio.h>

#include <stdlib.h>

struct Node

{

  int data;

  struct Node *next;

};

void push(struct Node** head_ref, int new_data)

{

struct Node* new_node = (struct Node*)
malloc(sizeof(struct Node));

new_node->data  = new_data;

new_node->next = (*head_ref);

(*head_ref)    = new_node;

}
```

```
void insertAfter(struct Node* prev_node, int
new_data)
{
if (prev_node == NULL)
    {
      printf("the given previous node cannot be
NULL");
      return;
    }
struct Node* new_node =(struct Node*)
malloc(sizeof(struct Node));
new_node->data  = new_data;
new_node->next = prev_node->next;
prev_node->next = new_node;
}
void append(struct Node** head_ref, int new_data)
{
struct Node* new_node = (struct Node*)
malloc(sizeof(struct Node))
struct Node *last = *head_ref;
new_node->data  = new_data;
new_node->next = NULL;
if (*head_ref == NULL)
    {
       *head_ref = new_node;
       return;
    }
while (last->next != NULL)
        last = last->next;
 last->next = new_node;
    return;
}
void printList(struct Node *node)
{
  while (node != NULL)
  {
     printf(" %d ", node->data);
     node = node->next;
  }
}
  int main()
{
 struct Node* head = NULL;
 append(&head, 6);
 push(&head, 7);
 push(&head, 1);
 append(&head, 4);
 insertAfter(head->next, 8);
 printf("\n Created Linked list is: ");
  printList(head);
  return 0;
}
```

Question 2.

Write a function "delete_beg()" for deleting a node from the beginning of the linked list.

Sol:

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
    int data;
    struct Node *next;
};
void push(struct Node** head_ref, int new_data)
{
    struct Node* new_node = (struct Node*)
malloc(sizeof(struct Node));
    new_node->data  = new_data;
    new_node->next = (*head_ref);
    (*head_ref)    = new_node;
```

```
}
void deleteNode(struct Node **head_ref, int key)
{
 struct Node* temp = *head_ref, *prev;
 if (temp != NULL && temp->data == key)
    {
        *head_ref = temp->next;
        free(temp);                       return;
    }
 while (temp != NULL && temp->data != key)
    {
        prev = temp;
        temp = temp->next;
    }
 if (temp == NULL) return;
 prev->next = temp->next;
 free(temp);
  void printList(struct Node *node)
{
    while (node != NULL)
    {
        printf(" %d ", node->data);
        node = node->next;
    }
}
 int main()
{
  struct Node* head = NULL;

    push(&head, 7);
    push(&head, 1);
    push(&head, 3);
    push(&head, 2);

    puts("Created Linked List: ");
    printList(head);
    deleteNode(&head, 1);
    puts("\nLinked List after Deletion of 1: ");
    printList(head);
    return 0;
}
```

Question 3.

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

Sol:

```
#include<stdio.h>
#include<stdlib.h>
void create(int);
void end_delete();
struct node
{
    int data;
    struct node *next;
};
struct node *head;
void main ()
{
    int choice,item;
    do
    {
        printf("\n1.Append List\n2.Delete node\n3.Exit\n4.Enter your choice?");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            printf("\nEnter the item\n");
            scanf("%d",&item);
            create(item);
            break;
            case 2:
            end_delete();
            break;
            case 3:
            exit(0);
            break;
            default:
            printf("\nPlease enter valid choice\n");
        }

    }while(choice != 3);
}
void create(int item)
```

```
    {
        struct node *ptr = (struct node
 *)malloc(sizeof(struct node *));
        if(ptr == NULL)
        {
            printf("\nOVERFLOW\n");
        }
        else
        {
            ptr->data = item;
            ptr->next = head;
            head = ptr;
            printf("\nNode inserted\n");
        }

    }
  void end_delete()
    {
        struct node *ptr,*ptr1;
        if(head == NULL)
        {
            printf("\nlist is empty");
        }
        else if(head -> next == NULL)
        {
            head = NULL;
            free(head);
            printf("\nOnly node of the list
deleted ...");
        }

        else
        {
            ptr = head;
            while(ptr->next != NULL)
                {
                    ptr1 = ptr;
                    ptr = ptr ->next;
                }
                ptr1->next = NULL;
                free(ptr);
                printf("\n Deleted Node from the
last ...");
            }
        }
```