Name - Chiradeep Banik

Enroll. No. - 20UCS176

Sec - A

Q.1

Linked List	Linear Array
• Memory is allocated dynamically.	• Memory is staticly allocated.
• Accessing any element is O(n) ~~com~~ time complexity	• Accessing any element is of O(1) time complexity.

We use array

↳ when we have large set of known data. and require fast data accessing at any index.

We use linked-list

↳ when we have a set of constantly changing size of data and they are linked to one-another.

→

Q.2

```
void insert_before_particular(struct linked_list *list,
                                      int data, int val){

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

   new_node->data = data;
   new_node->next = NULL;
   struct node *temp = list->head;
   if (list->head->data == val){
        new_node->next = list->head;
        list->head = new_nod;
        list->length++;
        printf("Success !!\n");
        return;
   }
   while (temp->next != NULL && temp->next->data != val){
        temp = temp->next;
   }
   if (temp->next == NULL){
      printf("the element is absent\n");
      return;
   }
   new_node->next = temp->next;
```

⟶

```
temp->next = new_node;
list->length++;
printf("Success !!\n");
}
```

Q.3

To delete a loop, we need to have two pointer, slow and fast. If they converge at any point then there is a loop in the list.

```
-> void detect_loop(struct linked_list *list){
       struct node *slowptr = list->head;
       struct node* fast_ptr= list->head;
       while (fast_ptr !=NULL && fast_ptr->next !=NULL){
           slow_ptr = slow_ptr->next;
           fast_ptr = fast_ptr->next->next;
           if (slow_ptr == fast_ptr){
               printf("Loop detected !!\n");
               return;
           }
       }
       printf("No loop detected !!\n");
   }
```

Q.4

```
→ void ~~dete~~ remove_loop (struct linked_list *list){

    struct node* slow = list→head ;
    struct node* fast = list→head;
    while (fast→next != NULL || fast != NULL){
        static int loop = 0;
        slow = slow→next;
        fast = fast→next→next;
        if (slow == fast){
            loop = 1;
            break;
        }
    }
    if (loop==0){
        return;
    }
    struct node* p1 = list→head;
    while(p1→next != slow→next){
        P1 = P1→next;
        slow = slow→next;
    }
    p1→next = NULL;
    printf("Removed Loop!!\n");
}
```

Q.5

```
→ void delete_n_after_m(struct linked_list *list, int n,
                                                   int m){
      struct node* current = list->head;
      struct node* prev = NULL;
      int count = 0;
      while (count < m && m != 0){
            prev = current;
            current = current->next;
            count++;
      }
      while (count < m+n){
            current = current->next;
            count++;
      }
      prev->next = current;
      list->length -= n;
      printf("Success!! \n");
  }
```

Q.6

Advantages of Lineaa linked List

- Dynamic memory allocation
- Constant time removal of head node.

Dis advantages

- Lineaa time data accessing
- Working with NULLs.