29/1/2024

CONCATENATION:

```
void concatenate (struct node * a,
struct node * b)
{
  if (a != NULL && b != NULL)
  {
    if (a -> next == NULL)
      a -> next = b;
    else
    concatenate (a -> next, b);
  }
else
  {
    printf (" Either a or b is NULL");
  }
```

struct node * concat (struct node * start 1,
struct node * start 2);

```
}
struct node * ptr;
if (start1 == NULL)
{
    start1 = start2;
    return start1;
}
if (start2 == NULL)
        return start1;
ptr = start1;
while (ptr -> link != NULL)
        ptr = ptr -> link;
    ptr -> link = start2;
return start1;
}
```

O/P:

First Linked list: 1 2 3
Second Linked list: 4 5
Concatenated linked list: 1 2 3 4 5

DOMS Page No. Date / /

REVERSING

```
struct node * prev = NULL;
struct node * current = head;
struct node * next = NULL
while (current != NULL)
  {
   next = current -> next;
   current -> next = prev;
   current = next;
  }
* head_ref = prev;
}
struct node * reverse (struct node * head){
```

O/P:

Original linked list: 5 4 3 2 1
Reversed linked list: 1 2 3 4 5

SORTING:

```
void insertion sort (struct node * head)
{
  struct node * sorted = NULL;
  struct node * current = head;
  while (current != NULL){
    struct Node * next = current -> next;
    sorted Insert (sorted, current);
    current = next;
  }
head = sorted;
}
```

OIP:

Original linked list: 5 4 8 1 6

Sorted linked list: 1 4 5 6 8

DOMS | Page No.
Date / /

⇒ STACK IMPLEMENTATION USING SINGLY LL

```
struct node * push (struct node *head, int value)
{
struct Node * temp = (struct node *) malloc
(size of (struct Node));
temp → data = value;
temp → next = head;
head = temp;
return head;
}

struct node * pop (struct node * head)
{
if (head == NULL)
  printf ("Stack is empty \n");
  return head;
}
```

```
struct node * temp = head;
head = temp -> next;
free (temp);
return head;
}
void display (struct Node * head) {
    struct node * d = head;
```

05/02/24

Double linked list: (a) create list
(b) display list (c) insert at the beginning
(d) delete node

```
# include <stdio.h>
# include <stdlib.h>
  struct node {
      struct node * next;
      int data;
      struct node * prev;
  };
struct node * start = NULL;
struct node * create_ll (struct node*);
struct node * display (struct node*)
struct node * insert_beg(struct node*)
struct node * insert_before (struct node*)
struct node * delete_node (struct node*)

struct node * create_ll (struct node * start)
```

```
{
struct node * newnode, * ptr;
int num;
printf (" \n enter -1 to end");
printf (" \n enter a value");
while (num != -1)
 {
   if (start == NULL)
   {
newnode = (struct node*) malloc (sizeof
(struct node));
newnode → prev = NULL;
newnode → data = NULL;
newnode → next = NULL;
start = newnode;
3
  else
  {
    ptr = start;
```

```
newnode = (struct node*) malloc (size of
(struct node));
newnode -> data = num;
while (ptr -> next != NULL)
  {
   ptr = ptr -> next;
  }
  ptr -> next = newnode;
  newnode -> prev = ptr;
  newnode -> next = NULL;
  }
  printf ("\n Enter the data:");
  scanf (" %d", &num);
  }
  return start;
};

struct node * display (struct node * start)
{
```

```
struct node * ptr;
ptr = start;
while (ptr != NULL)
 {
  printf ("\t %d", ptr -> data);
  ptr = ptr -> next;
 }
return start;
}

struct node * insert beg (struct node * start)
 {
  struct node * newnode;
  int num;
  printf ("\n Enter the data");
  scanf ("%d", &num);
  newnode = (struct node *) malloc (sizeof
  newnode -> data = num
  start prev = newnode;
```

```
newnode -> next = start;
newnode -> prev = NULL;
start = newnode;
return start;
};

struct node * insert before (struct node *
{
struct node *ptr, *newnode;
int num, val;
printf ("\n Enter the data");
scanf ("%d", &num);
printf ("\n Enter value before which");
scanf ("%d", val);
newnode = (struct node *) malloc (sizeof
(struct node));
newnode -> data = num;
ptr = start;
while (ptr -> data != val)
{
```

```
ptr -> ptr -> next;
}
newnode -> next = ptr;
newnode -> prev = ptr -> prev
ptr -> prev -> next = newnode;
ptr -> prev = newnode;
return start;
};


struct node * delete_node (struct node *
start)
{

struct node * ptr;
int val
printf ("Enter value to be deleted");
scanf ("%d", &val);
ptr = start;
while (ptr -> data != val)

    ptr = ptr -> next

ptr -> prev -> next = ptr -> next;
ptr -> next -> prev = ptr -> prev;
```

```
free (ptr);
return start;
}

int main ()
 {
  int choice
  printf ("Menu")
  printf ("1. create List")
  printf (" 2. Insert before");
  printf (" 3. Delete");

printf ("4. display");
do {
   printf ("Enter your choice");
   scanf ("%d", & choice);
   switch (choice)
   {

   case 1: start = create - list (start);
           break;
```

```
case 2: start = insert_begin(start);
        break
case 3: start = delete_node(start);
        break;
case 4: start = display(start);
        break
} while(choice != 5);
}
```