25/01/24

Date __/__/__
Page _____

Lab-5

① Linked list

```
#include <stdio.h>
#include <stdlib.h>

struct Node
{
    int data;
    struct Node *next;
};
typedef struct Node Node;
Node * createNode (int value)
{
    Node *newnode = (Node *) malloc (sizeof (Node));
    newNode -> data = value;
    newNode -> next = NULL;
    return newNode;
}

void display (Node *head)
{
    while (head != NULL)
    {
        printf ("%d -> ", head -> data);
        head = head -> next;
    }
    printf ("NULL \n");
}

Node * sortlist (Node * head)
{
    if (head == NULL || head -> next == NULL)
        return head;
```

Emp LL 25/1/24

```
int swapped;
Node * temp;
Node * end = NULL;
do
{
swapped = 0;
temp = head;
while (temp -> next != end)
{
if (temp -> data > temp -> next -> data)
{
int tempData = temp -> data;
temp -> data = temp -> next -> data;
temp -> next -> data = tempData;
swapped = 1;
}
temp = temp -> next;
}
end = temp;
} while (swapped);
return head;
}

Node * reverselist (Node *head)
{
Node * prev = NULL;
Node * curr = head;
Node * nextNode = NULL;
while (curr != NULL)
{
nextNode = curr -> next;
curr -> next = prev;
```

```
        prev = curr;
        curr = nextNode;
    }
    return prev;
}

Node * concatLists(Node *list1, Node *list2)
{
    if (list1 == list2)
        return list2;
    Node *temp = list1;
    while (temp -> next != NULL)
    {
        temp = temp -> next;
    }
    temp -> next = list2;
    return list1;
}
void main()
{
    Node * list1 = CreateNode(18);
    list1 -> next = CreateNode(7);
    list1 -> next -> next = CreateNode(45);

    Node * list2 = CreateNode(17);
    list2 -> next = CreateNode(333);

printf("original list 1:");
display(list1);
printf("original list 2:");
display(list1);
```

```
Node *concatenated = concatLists(list1, list2);
printf("Concatenated List: ");
display(concatenated);
}
```

Output:

Original list 1: 18 → 7 → 45 → NULL
Original list 2: 17 → 333 → NULL
Sorted list = 7 → 18 → 45 → NULL
Reversed list: 45 → 18 → 7 → NULL
Concatenated list: 45 → 18 → 7 → 17 → 333 → NULL

Stack :-

typedef

} Stack

void p

}

int P

{

}

}

uo

Date / /
Page

Date / /
Page

Stack :-

```
typedef struct {
    node * top;
} LinkedList;

void push (LinkedList * Stack, int value) {
    Node * newNode = createNode(value);
    newNode -> next = Stack -> top;
    Stack -> top = newNode;
}

int pop (LinkedList * stack)
{
    if (Stack -> top == NULL) {
        printf ("Stack is Empty \n");
        return -1;
    }
    int poppedvalue = Stack -> top -> data;
    Node * temp = stack -> top;
    Stack -> top = Stack -> top -> next;
    free (temp);
    return poppedvalue;
}

void main () {
    LinkedList stack;
    Stack.top = NULL;
    printf (" Stack Operation : \n");
    push (&stack, 18);
    push (& Stack, 7);
    push (& stack, 17);
    push (&Stack, 45);
    display (Stack.top);
```

25/11/24

Date __/__/__
Page _____

printf("popped value = %d \n", pop(&stack));
printf("popped val : %d \n", pop(&stack));
display(stack.top);

}

Output :-

Stack operation:
45 → 17 → 7 → 18 → NULL
popped value : 45
popped value : 17
7 → 18 → NULL

Queue :-

void enqueu
{
node *
if (que
{
qu
que
}
else
{
qu
q
}
}

int dequeu
{
if (
{

}
int
No
que
free
ret
}

Date / /
Page

Queue :-

```
void enqueue (LinkedList *queue, int value)
{
    Node * newNode = CreateNode (value);
    if (queue -> front == NULL)
    {
        queue -> front = newNode;
        queue -> rear = newNode;
    }
    else
    {
        queue -> rear -> next = newNode;
        que -> rear = newNode;
    }
}

int dequeue (LinkedList * queue)
{
    if (queue -> front == NULL)
    {
        printf ("queue is empty : \n");
        return -1;
    }
    int dequeuevalue = queue -> front -> data;
    Node * temp = queue -> front;
    queue -> front = queue -> front -> next;
    free (temp);
    return dequeuevalue;
}
```

```
void main()
{
    linkedlist queue;
    queue.front = NULL;
    queue.rear = NULL;

    printf("\n queue operations : \n");
    enqueue(&queue, 18);
    enqueue(&queue, 7);
    enqueue(&queue, 45);
    display(queue.front);
    printf("dequeued from queue : %d \n", dequeue
                                        (&queue));
    printf("dequed from queue : %d \n", dequeue
                                        (&queue));
    display(queue.front);
}
```

25/11/24

Output :-

queue operations :
18 → 7 → 45 → NULL
dequed from que = 18
dequed from queue = 7
45 → NULL