LINKED LISTS

BY ASRITHA MEKA, AP19110010224, CSE-G

41)

Objective:

Write a C program to create a single linked list with 5 nodes.

Code:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node{
        int data;
        struct Node *next;
}node;
node* createNode(int data){
        node *a = ((node*)malloc(sizeof(node)));
        a->data = data;
        a->next = NULL;
        return a;
}
node* createList(){
        int a=8,data;
        node  *x, *head = NULL;
        while(a--){
        printf("Enter the number\n");
        scanf("%d",&data);
        if(head == NULL){
        head = createNode(data);
        x = head;
        }
        else{
        x->next = createNode(data);
        x = x->next;
        }
        }
        return head;

}
void display(node *head){
        while(head!=NULL){
        printf("%d->",head->data);
        head = head->next;
        }
        printf("NULL\n");
}
int main()
{
        node *head=createList();
        display(head);
        return 0;
}
```

Output:

```
Enter the number
45
Enter the number
67
Enter the number
76
Enter the number
 98
```

42)

Objective:

Write a c program to search an element in a singly linked list.

Code:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node{
```

```
            int data;
            struct Node *next;
}node;
node* createNode(int data){
            node *a = ((node*)malloc(sizeof(node)));
            a->data = data;
            a->next = NULL;
            return a;
}
node* createList(){
            int a,data;
            node  *x, *head = NULL;
            printf("\n to enter how many elements?");
            scanf("%d", &a);
            while(a--){
            printf("Enter the number\n");
            scanf("%d",&data);
            if(head == NULL){
            head = createNode(data);
            x = head;
            }
            else{
            x->next = createNode(data);
            x = x->next;
            }
            }
            return head;

}
int search(node *head,int search){
 int pos=1;
 while(head != NULL){
  if(head->data == search) return pos;
  head = head->next;
  pos++;
 }
 return -1;
}
int main(int argc, char const *argv[])
{
 node *head = createList();
 int s;
 printf("element to be a searched\n");
 scanf("%d",&s);
 printf("found at the %d\n",search(head,s));
 return 0;
}
```

Output:

```
 to enter how many elements?
45
Enter the number
66
Enter the number
76
Enter the number
54
Enter the number
76
```

43)

Objective:

Write a c program to perform the following takes:

1. Insert a node at the beginning of a singly-linked list.
2. Insert a node at beginning of a singly - linked list.
   ……….

Code:

```
#include <stdio.h>
#include <stdlib.h>
#define init() ((struct node*)malloc(sizeof(struct node)))
typedef struct node
{
 int data;
 struct node *next;
 struct node *back;
```

```
}node;
node* createNode(int data){
            node *a = ((node*)malloc(sizeof(node)));
            a->data = data;
            a->next = NULL;
            return a;
}
node* insertBeg(node *head,int data){
 node *newNode = createNode(data);
 newNode->next = head;
 return newNode;
}
void insertMiddle(node *head,int data){
 node *ptr = head;
 if(ptr == NULL){
  printf("empty\n");
  return;
 }
 while(head->next != NULL){
  if(head->next->next != NULL){
   head = head->next->next;
   ptr = ptr->next;
  }
  else{
   break;
  }
 }
 node *temp = ptr->next;
 ptr->next = createNode(data);
 ptr->next->next = temp;
}
node* insertEnd(node *head, int data){
 if(head == NULL){
  return createNode(data);
 }
 head->next = insertEnd(head->next, data);
 return head;

}
node* deleteBeg(node *head){
 node *temp = head;
 if(head == NULL){
  printf("Empty\n");
  return NULL;
 }
 printf("%d deleted\n",temp->data);
 free(temp);
 return head->next;
}
node* deleteEnd(node *head){
 if(head == NULL ){
  printf("empty\n");
  return NULL;
 }
 if (head->next == NULL){
  printf("%d deleted\n",head->data );
  free(head);
  return NULL;
 }
 head->next = deleteEnd(head->next);
 return head;
}
void display(node *head){
	while(head!=NULL){
	printf("%d->",head->data);
	head = head->next;
	}
	printf("NULL\n");
}
int main () {
 int choice,data;
 node *head ;
	while(1){
	printf("\n***Main Menu*\n");
	printf("\nChoose one option from the list ...\n");
	printf("\n===============================\n");
	printf("\n1.Insert in the begining\n2.Insert at the last\n3.Insert at the middle.\n4.Delete
the num at the begining \n5.Delete num at the end\n6.Display\n7.Exit\n");
	printf("\nEnter your choice?\n");
	scanf("\n%d",&choice);
	switch(choice){

	case 1:{
	 printf("Enter the data for inserted\n");
	 scanf("%d",&data);
	 head = insertBeg(head, data);
```

```
            break;
           }
           case 2:{
            printf("Enter the data for inserted\n");
            scanf("%d",&data);
            head = insertEnd(head, data);
            break;
           }
           case 3:{

            printf("Enter the data for inserted\n");
            scanf("%d",&data);
            insertMiddle(head, data);
            break;
           }

           case 4:{
            head = deleteBeg(head);
            break;
           }
           case 5:{
            head = deleteEnd(head);
            break;
           }
            case 6:{
             printf("The list:\n");
             display(head);
             break;
            }
           case 7: {exit(0);break;}
           }
           }
 return 0;
}
```

output:

```
***Main Menu*
Choose one option from the list ...
==============================
1.Insert in the begining
```

```
2.Insert at the last
3.Insert at the middle.
4.Delete the num at the begining
5.Delete num at the end
6.Display
7.Exit
Enter your choice?
45
***Main Menu*
Choose one option from the list ...
===============================
1.Insert in the begining
2.Insert at the last
3.Insert at the middle.
4.Delete the num at the begining
5.Delete num at the end
6.Display
7.Exit
Enter your choice?
4
Empty
***Main Menu*
Choose one option from the list ...
===============================
1.Insert in the begining
2.Insert at the last
3.Insert at the middle.
4.Delete the num at the begining
5.Delete num at the end
6.Display
7.Exit
Enter your choice?
4
Empty
***Main Menu*
Choose one option from the list ...
===============================
1.Insert in the begining
2.Insert at the last
3.Insert at the middle.
4.Delete the num at the begining
5.Delete num at the end
6.Display
```

```
7.Exit
Enter your choice?
6
The list:
NULL
***Main Menu*
Choose one option from the list ...
===============================
1.Insert in the begining
2.Insert at the last
3.Insert at the middle.
4.Delete the num at the begining
5.Delete num at the end
6.Display
7.Exit
Enter your choice?
4
```

44)

Objective:

Write a c program to create a doubly linked list with 5 nodes.

Code:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node
{
 int data;
 struct node *next;
 struct node *back;
}node;
node* createNode(int data){
 node *newNode = ((node*)malloc(sizeof(node)));
 newNode->data = data;
 newNode->next = NULL;
 newNode->back = NULL;
 return newNode;
}
node* createList(){
```

```
            int n=8,data;
            node  *x, *head = NULL,*temp;
            while(n--){
            printf("Enter the number\n");
            scanf("%d",&data);
            if(head == NULL){
            head = createNode(data);
            x = head;
            }
            else{
            temp = createNode(data);
            x->next = temp;
            temp->back= x;
            x = x->next;
            }
            }
            return head;

}
void display(node *head){
 while(head != NULL){
  printf("%d ",head->data);
  head = head->next;
 }
 printf("\n");
}
int main(){
 node *head = createList();
 display(head);
 return 0;
}
```

Output:

```
Enter the number
 65
Enter the number
87
Enter the number
87
Enter the number
78
Enter the number
```

```
46
Enter the number
53
65 87 87 78 46 53
```

46)

Objective:

Write a C program to implement the stack using linked lists.

Code:

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

struct node
{
	int info;
	struct node *ptr;
}*x,*x1,*temp;

int topelement();
void push(int data);
void pop();
void empty();
void display();
void destroy();
void stack_count();
void create();

int count = 0;

void main()
{
	int no, ch, e;

	printf("\n 1 - Pop the element");
	printf("\n 2 - Push the element ");
	printf("\n 3 - Top of element");
	printf("\n 4 - Empty set");
printf("\n 5 - Exit");
printf("\n 6 - Dipslay");
printf("\n 7 - Stack Counting");
printf("\n 8 - Destroy the stack");

create();

while (1)
{
printf("\n Enter the choice : ");
scanf("%d", &ch);

switch (ch)
{
case 1:
printf("Enter the data : ");
scanf("%d", &no);
push(no);
break;
case 2:
pop();
break;
case 3:
if (x == NULL)
        printf("No elements in stack");
else
{
        e = topelement();
        printf("\n Top element : %d", e);
}
break;
case 4:
empty();
break;
case 5:
exit(0);
case 6:
display();
break;
case 7:
stack_count();
break;
case 8:
        destroy();
        break;
        default :
        printf(" it is incorrect, Please enter correct choice  ");
        break;
        }
        }
}

void create()
{
        x = NULL;
}

void stack_count()
{
        printf("\n No. of elements in stack : %d", count);
}

void push(int data)
{
        if (x == NULL)
        {
        x =(struct node *)malloc(1*sizeof(struct node));
        x->ptr = NULL;
        x->info = data;
        }
        else
        {
        temp =(struct node *)malloc(1*sizeof(struct node));
        temp->ptr = x;
        temp->info = data;
        x = temp;
        }
        count++;
}

void display()
{
        x1 = x;

        if (x1 == NULL)
        {
```

```
        printf("Stack is empty");
        return;
        }

        while (x1 != NULL)
        {
        printf("%d ", x1->info);
        x1 = x1->ptr;
        }
 }

void pop()
{
        x1 = x;

        if (x1 == NULL)
        {
        printf("\n Error : Trying to pop from empty stack");
        return;
        }
        else
        x1 = x1->ptr;
        printf("\n Popped value : %d", x->info);
        free(x);
        x = x1;
        count--;
}

int topelement()
{
        return(x->info);
}

void empty()
{
        if (x == NULL)
        printf("\n Stack is empty");
        else
        printf("\n Stack is not empty with the %d elements", count);
}

void destroy()
{
```

```
            x1 = x;

            while (x1 != NULL)
            {
            x1 = x->ptr;
            free(x);
            x = x1;
            x1 = x1->ptr;
            }
            free(x1);
            x = NULL;

            printf("\n All stack elements are to be destroyed");
            count = 0;
}
```

Output:

```
 1 - Pop the element
 2 - Push the element
 3 - Top of element
 4 - Empty set
 5 - Exit
 6 - Dipslay
 7 - Stack Counting
 8 - Destroy the stack
 Enter the choice : 4
 Stack is empty
 Enter the choice : 6
Stack is empty
 Enter the choice : 7 3
No elements in stack
 Enter the choice :
```

47)

Objective:

Write a C program to implement the queue using a linked list.

Output:

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

struct node
{
	int info;
	struct node *ptr;
}*x,*rear,*temp,*x1;

int frontelement();
void enq(int data);
void deq();
void empty();
void display();
void create();
void queuesize();

int count = 0;

void main()
{
	int no, ch, e;

	printf("\n 1 - Enque");
	printf("\n 2 - Deque");
	printf("\n 3 - Front element");
	printf("\n 4 - Empty");
	printf("\n 5 - Exit");
	printf("\n 6 - Display");
	printf("\n 7 - Queue size");
	create();
	while (1)
	{
	printf("\n Enter the choice : ");
	scanf("%d", &ch);
	switch (ch)
	{
	case 1:
	printf("Enter the data : ");
	scanf("%d", &no);
```

```
        enq(no);
        break;
        case 2:
        deq();
        break;
        case 3:
        e = frontelement();
        if (e != 0)
                printf("Front element : %d", e);
        else
                printf("\n No front element in the Queue is empty");
        break;
        case 4:
        empty();
        break;
        case 5:
        exit(0);
        case 6:
        display();
        break;
        case 7:
        queuesize();
        break;
        default:
        printf("incorect choice, Please enter correct choice  ");
        break;
        }
        }
}

void create()
{
        x = rear = NULL;
}

void queuesize()
{
        printf("\n Queue size : %d", count);
}

void enq(int data)
{
        if (rear == NULL)
        {
        rear = (struct node *)malloc(1*sizeof(struct node));
        rear->ptr = NULL;
        rear->info = data;
        x = rear;
        }
        else
        {
        temp=(struct node *)malloc(1*sizeof(struct node));
        rear->ptr = temp;
        temp->info = data;
        temp->ptr = NULL;

        rear = temp;
        }
        count++;
}

void display()
{
        x1 = x;
        if ((x1 == NULL) && (rear == NULL))
        {
        printf("Queue is empty");
        return;
        }
        while (x1 != rear)
        {
        printf("%d ", x1->info);
        x1 = x1->ptr;
        }
        if (x1 == rear)
        printf("%d", x1->info);
}

void deq()
{
        x1 = x;

        if (x1 == NULL)
        {
        printf("\n Error: Trying to display the elements from the empty queue");
        return;
        }
        else
        if (x1->ptr != NULL)
        {
        x1 = x1->ptr;
        printf("\n Dequed value : %d", x->info);
        free(x);
        x = x1;
        }
        else
        {
        printf("\n Dequed value : %d", x->info);
        free(x);
        x = NULL;
        rear = NULL;
        }
        count--;
}

int frontelement()
{
        if ((x != NULL) && (rear != NULL))
        return(x->info);
        else
        return 0;
}

void empty()
{
        if ((x == NULL) && (rear == NULL))
        printf("\n Queue is empty");
        else
        printf("Queue is not empty");
}
```

Output:

```
 1 - Enque
 2 - Deque
 3 - Front element
 4 - Empty
 5 - Exit
```

```
6 - Display
7 - Queue size
Enter the choice : 5
```