DS LAB

NAME: SOUMYADIP MAITY

ROLL NO.: 22053029

SECTION: CSE 49

YEAR: 2022-23

ASSIGNMENT- 4

DATE: 01.09.2023

Q1: Write a menu-driven program to perform the following operations in asingle linked list by
using suitable defined functions for each case.
a) Tranversal of the list
b)Check if the list is empty
c)Insert a node at the certain position
d)Delete a node at the certain position
e)Delete a node for the given key
f)Count a node for the given key
g)Search for an element in the linked list

```
#include <stdio.h>
#include <stdlib.h>
struct node {
int info;
struct node* link;
};
struct node* start = NULL;
void createList()
{
if (start == NULL) {
int n;
printf("\nEnter the number of nodes: ");
scanf("%d", &n);
if (n != 0) {
int data;
struct node* newnode;
struct node* temp;
newnode = malloc(sizeof(struct node));
start = newnode;
temp = start;
printf("\nEnter number to"
" be inserted : ");
scanf("%d", &data);
start->info = data;
for (int i = 2; i <= n; i++) {
newnode = malloc(sizeof(struct node));
temp->link = newnode;
printf("\nEnter number to"
" be inserted : ");
scanf("%d", &data);newnode->info = data;
temp = temp->link;
}
```

```
}
printf("\nThe list is created\n");
}
else
printf("\nThe list is already created\n");
}
void traverse()
{
struct node* temp;
if (start == NULL)
printf("\nList is empty\n");
else {
temp = start;
while (temp != NULL) {
printf("Data = %d\n", temp->info);
temp = temp->link;
}
}
}
void insertAtFront()
{
int data;
struct node* temp;
temp = malloc(sizeof(struct node));
printf("\nEnter number to"
" be inserted : ");
scanf("%d", &data);
temp->info = data;
temp->link = start;
start = temp;
}
void insertAtEnd()
{
int data;
struct node *temp, *head;
temp = malloc(sizeof(struct node));
printf("\nEnter number to"
" be inserted : ");
scanf("%d", &data);
temp->link = 0;
temp->info = data;
head = start;
while (head->link != NULL) {
```

```
head = head->link;
}head->link = temp;
}
void insertAtPosition()
{
struct node *temp, *newnode;
int pos, data, i = 1;
newnode = malloc(sizeof(struct node));
// Enter the position and data
printf("\nEnter position and data :");
scanf("%d %d", &pos, &data);temp = start;
newnode->info = data;
newnode->link = 0;
while (i < pos - 1) {
temp = temp->link;
i++;
}
newnode->link = temp->link;
temp->link = newnode;
}
void deleteFirst()
{
struct node* temp;
if (start == NULL)
printf("\nList is empty\n");
else {
temp = start;
start = start->link;
free(temp);
}
}
void deleteEnd()
{
struct node *temp, *prevnode;
if (start == NULL)
printf("\nList is Empty\n");
else {
temp = start;
while (temp->link != 0) {
prevnode = temp;
temp = temp->link;
}
free(temp);
```

```
prevnode->link = 0;
}}
void deletePosition()
{
struct node *temp, *position;
int i = 1, pos;
if (start == NULL)
printf("\nList is empty\n");
else {
printf("\nEnter index : ");
scanf("%d", &pos);
position = malloc(sizeof(struct node));
temp = start;
while (i < pos - 1) {
temp = temp->link;
i++;
}
position = temp->link;
temp->link = position->link;
free(position);
}
}
void maximum()
{
int a[10];
int i;
struct node* temp;
if (start == NULL)
printf("\nList is empty\n");
else {
temp = start;
int max = temp->info;
while (temp != NULL) {
if (max < temp->info)
max = temp->info;
temp = temp->link;
}
printf("\nMaximum number "
"is : %d ",
max);
}
}
void search()
```

```
{
int found = -1;
struct node* tr = start;
if (start == NULL) {
printf("Linked list is empty\n");
}
else {
printf("\nEnter the element you want to search: ");
int key;
scanf("%d", &key);
while (tr != NULL) {
if (tr->info == key) {
found = 1;
break;
}
else {
tr = tr->link;
}
}
 if (found == 1) {
printf(
"Yes, %d is present in the linked list.\n",
key);
}
else {
printf("No, %d is not present in the linked "
"list.\n",
key);
}
}}
int main()
{
int choice;
while (1) {
printf("\n\t1 To see list\n");
printf("\t2 For insertion at"
" starting\n");
printf("\t3 For insertion at"
" end\n");
printf("\t4 For insertion at "
"any position\n");
printf("\t5 For deletion of "
"first element\n");
```

```
printf("\t6 For deletion of "
"last element\n");
printf("\t7 For deletion of "
"element at any position\n");
printf("\t8 To find maximum among"
" the elements\n");
printf("\t12 Search an element in linked list\n");
printf("\t13 To exit\n");
printf("\nEnter Choice :\n");
scanf("%d", &choice);
switch (choice) {
case 1:
traverse();
break;
case 2:
insertAtFront();
break;
case 3:
insertAtEnd();
break;
case 4:
insertAtPosition();
break;
case 5:
deleteFirst();
break;
case 6:
deleteEnd();
break;
case 7:
deletePosition();
break;
case 8:
maximum();break;
case 9:search();
break;
case 10:
exit(1);
break;
default:
printf("Incorrect Choice\n");
}
}
```

```
return 0;
}
}
```

Q2: WAP to search an element in a simple linked list , if found delete that node and insert that node at beginning

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int num;
struct node *nextptr;
}
stnode, *ennode;
int FindElement(int);
void main()
{
int n,i,FindElem,FindPlc;
stnode.nextptr=NULL;
ennode=&stnode;
printf("\n\n Linked List : Search an element in a Singly Linked List :\n");
printf("-----------------------------------------------------------------\n");
printf(" Input the number of nodes : ");
scanf("%d", &n);
printf("\n");
for(i=0;i< n;i++)
{
ennode->nextptr=(struct node *)malloc(sizeof(struct node));
printf(" Input data for node %d : ",i+1);
scanf("%d",&ennode->num);
ennode=ennode->nextptr;}
ennode->nextptr=NULL;
printf("\n Data entered in the list are :\n");
ennode=&stnode;
while(ennode->nextptr!=NULL)
{
printf(" Data = %d\n",ennode->num);
ennode=ennode->nextptr;
}
printf("\n");
printf(" Input the element to be searched : ");
scanf("%d",&FindElem);
```

```
FindPlc=FindElement(FindElem);
if(FindPlc<=n)
printf(" Element found at node %d \n\n",FindPlc);
else
printf(" This element does not exists in linked list.\n\n");
}
int FindElement(int FindElem)
{
int ctr=1;
ennode=&stnode;while(ennode->nextptr!=NULL)
{
if(ennode->num==FindElem)
break;
else
ctr++;
ennode=ennode->nextptr;
}
return ctr;
```

Q3: WAP to count the number of occurances of an element in a linked list of n nodes

```
#include <stdio.h>
int occur(int [], int, int);
int main()
{
int size, key, count;
int list[20];
int i;
printf("Enter the size of the list: ");
scanf("%d", &size);
printf("Printing the list:\n");
for (i = 0; i < size; i++)
{
list[i] = rand() % size;
printf("%d ", list[i]);
}
printf("\nEnter the key to find it's occurence: ");
scanf("%d", &key);
count = occur(list, size, key);
printf("%d occurs for %d times.\n", key, count);
return 0;
```

```
}
int occur(int list[], int size, int key)
{
int i, count = 0;
for (i = 0; i < size; i++)
{
if (list[i] == key)
{
count += 1;
}
}
return count;
}
```

Q4: WAP to remove duplicates from a linked list of n nodes

```
#include<stdio.h>
#include<stdlib.h>
struct Node {
int data;
struct Node* next;
};
struct Node* newNode(int data)
{
struct Node* temp =
(struct Node*)malloc(sizeof(struct Node));temp->data = data;
temp->next = NULL;
return temp;
}
void removeDuplicates(struct Node* start)
{
struct Node *ptr1, *ptr2, *dup;
ptr1 = start;
while (ptr1 != NULL && ptr1->next != NULL) {
ptr2 = ptr1;
of the elements */
while (ptr2->next != NULL) {
if (ptr1->data == ptr2->next->data) {
dup = ptr2->next;
ptr2->next = ptr2->next->next;
free(dup);
}
else
```

```
ptr2 = ptr2->next;
}
ptr1 = ptr1->next;
}
}
void printList(struct Node* node)
{
while (node != NULL) {
printf("%d ", node->data);
node = node->next;
}
}
int main()
{
10->12->11->11->12->11->10*/
struct Node* start = newNode(10);
start->next = newNode(12);
start->next->next = newNode(11);
start->next->next->next = newNode(11);
start->next->next->next->next = newNode(12);
start->next->next->next->next->next = newNode(11);
start->next->next->next->next->next->next = newNode(10);
printf("Linked list before removing duplicates ");
printList(start);
removeDuplicates(start);
printf("\nLinked list after removing duplicates ");
printList(start);
return 0;
}
```