Lab ⇒ 29/01/2024

SURYA Gold

Date _____ Page _____

Q.) Write a program to sort, reverse and concatenate a linear linked list.

Ans⇒

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *next;
};
void insert_at_begin(struct node *head,
                     int data)
{
    struct node *newnode = (struct node *) malloc
                           (sizeof(struct node));
    newnode->data = data;
    newnode->next = head;
    head = newnode;
}
void printList(struct node *head) {
    while (head != NULL) {
        printf("%d", head->data);
        head = head->next;
    }
    printf("\n");
}
void sortList(struct node **head) {
    struct node *current, *nextnode;
    int temp;
```

```
current = *head;
while (current != NULL) {
    nextnode = current -> next;
    while (nextnode != NULL) {
        nextnode = current -> next;
        if (current -> data > nextnode -> data)
        {  temp = current -> data;
           current -> data = nextnode -> data;
           nextnode -> data = temp;
        }
        nextnode = nextnode -> next;
    }
    current = current -> next;
}

void reverseList (struct node ** headRef) {
    struct node *prev, *current, *nextnode;
    prev = NULL;
    current = *headRef;
    while (current != NULL) {
        nextnode = current -> next;
        current -> next = prev;
        prev = current;
        current = nextnode;
    }
    *headRef = prev;
}

void concatenateLists (struct node **list1,
    struct node *list2) {
```

```
if (*list1 == NULL) {
    *list1 = list2;
    return;
}
struct node *temp = *list1;
while (temp->next != NULL) {
        temp = temp->next;
}
temp->next = list2;
}
int main() {
    struct node *list1 = NULL;
    struct node *list2 = NULL;
    int choice;
    int data;
    printf("1. Insert into List 1 \n");
    printf("2. Insert into List 2 \n");
    printf("3. Sort List 1 \n");
    printf("4. Reverse List 1 \n");
    printf("5. Concatenate Lists \n");
    printf("6. Print Lists \n");
    printf("0. Exit \n");
while (1)
{
    printf("Enter choice: ");
    scanf("%d", &choice);
switch (choice) {
    case 1:
        printf("Enter data into list 1");
        scanf("%d", &data);
        insertatBeginning(&list1, data);
        break;
```

```
case 2:
    printf("Enter data to insert into list 2: ");
    scanf("%d", &data);
    insert_at_begin(&list2, data);
    break;

case 3:
    sortlist(&list1);
    printf("List 1 sorted");
    break;

case 4:
    reverselist(&list1);
    printf("List 1 reversed.\n");
    break;

case 5:
    ConcatenateLists(&list1, list2);
    printf("Lists concatenated\n");
    break;

case 6:
    printf("List 1\n");
    printList(list1);
    printf("List 2:");
    printList(list2);
    break;

case 7:
    printf("Exit"); exit(0);
    break;
```

```
    return 0;
}
```

Output :-

```
1. Insert into list 1
2. Insert into list 2
3. Sort List 1
4. Reverse List 2
5. Concatenate lists
6. Print lists
0. Exit

Enter your choice : 1
Enter data to insert into list 1 : 23
Enter your choice : 2
Enter data to insert into list 2 : 34
Enter your choice : 1
Enter data to insert into list 1 : 2
Enter data to insert into List 2 : 45
Enter your choice : 6
List 1 : 2 23
List 2 : 45 34
Enter your choice : 3
List 1 sorted.
Enter your choice : 4
List 1 reversed.
Enter your choice : 6
List 1 : 23 2
List 2 : 45 34
Enter your choice : 5
Lists concatenated.
Enter your choice : 6
```

List 1 : 23 2 45 34

List 2 : 45 34

Enter your choice : 0

Exit

Q.) Write a program to implement stack using linked list.

Ans)

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
};
struct node *top = NULL;
void push (int x) {

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

newnode -> data = x;
newnode -> next = top;
top = newnode;
}
void pop() {

    struct node *temp;
    temp = top;
if (top == 0) {
    printf("List is Empty");
}
```

```
else {
    top = top -> next;
    free(temp);
}
}

void main() {
    int choice = -1; int x;
    while (choice != 4)
    {
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Add at top:");
                scanf("%d", &x);
                push(x);
                break;
            case 2:
                printf("Pop from top");
                pop();
                break;

            case 3:
                printf("Display\n");
                display();
                break;

            case 4:
                printf("Exit");
                break;
        }
    }
}
```

```
void display()
{
    struct node * temp ;
    temp = top ;
    while (temp != NULL)
    {
        printf("%d ", temp -> data);
        temp = temp -> next ;
    }
}
```

Output :-

```
Enter choice : 1
Add at top : 42
Enter choice : 1
Add at top : 43
Enter choice : 3
43  42
Enter choice : 2
Pop from top
Enter choice : 3
42
Enter choice : 4
Exit
```

Q.) Queue implementation using linked list

Ans →

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node * next;
};
```

```
struct node *front = NULL, *rear = NULL;

void enqueue (int x)
{
    newnode = (struct node*) malloc (sizeof (struct node));
    newnode->data = x; newnode->next = NULL;
    if (head == NULL)
    {
        head = newnode;
    }
    else
    {
        temp = newnode;
    }
    if (front == 0 && rear == 0) {
        front = rear = newnode;
    }
    else {
        rear -> next = newnode;
        rear = newnode;
    }
}
void display () {
    struct node *temp;
    if (front == 0 && rear == 0) {
        printf ("Queue is empty");
    }
    else {
        temp = front;
        while (temp != NULL) {
            printf ("%d", temp -> data);
            temp = temp -> next;
        }
    }
}
```

```
void dequeue(){
    struct node *temp;
    temp = front;
if (front == 0 && rear == 0){
    printf("Empty queue !!");
}
else{
    front = front -> next;
    free(temp);
}
}

void main(){  int x;
    int choice = -1;
while (choice != 4)
{
    printf("Enter choice : ");
    scanf("%d", &choice);
  Switch (choice)
  {
      case 1:
      printf("Add element : ");
      scanf("%d", &x);
      enqueue(x);
      break;

      case 2:
      printf("Delete element : ");
      dequeue();
      break;
```

```
case 3:
        printf("Display");
        display();
        break;
case 4:
        printf("Exit");
        break;
    }
  }
}
```

Output -:

```
Enter choice : 1
Add element : 12
Enter choice : 1
Add element : 13
Enter choice : 1
Add element : 15
Enter choice : 3
Display
12 13 15
Enter choice : 2
Delete element
Enter choice : 3
13 15
Enter choice : 4
Exit
```

29.01.24