25/01/24 Week-6 : Linked List : Sort, Reverse, Concatenation

```
#include<stdio.h>
#include<stdlib.h>
struct Node{
  int data;
  struct Node * next;
};
struct Node * insertNode(struct Node * head,
int new_data){
  struct Node * new_node = (struct Node *)malloc
                          (sizeof(struct Node));
  new_node->data = new_data;
  new_node->next = head;
  return new_node;
}
struct Node* bubbleSort(struct Node *head)
{
  if(head==NULL || head->next==NULL)
  {
    return head;
  }
int swapped;
struct Node * temp;
struct 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;
}
struct Node * reverselist (struct Node * head)
{
  struct Node * prev = NULL;
  struct Node * current = head;
  struct Node * next = NULL;
  while (current != NULL)
  {
    next = current -> next;
    current -> next = prev;
    prev = current;
    current = next;
  }
  return prev;
}
struct Node * concatlist (struct Node * head1,
                          struct Node * head2)
{
  if (head1 == NULL)
  {
    return head2;
  }
  struct Node * temp = head1;
  while (temp != NULL)
  {
    printf ("%d", temp -> data);
    temp = temp -> next;
  }
  printf ("\n");
}
```

```
int main()
{
  struct Node * head = NULL;
  int choice, data;
  while(1)
  {
    printf("1. Insert node \n 2. sort linked list \n
            3. Reverse linked list \n 4. Concat node
            linked list \n 5. Print linked list \n
            6. Exit \n");
    printf("Enter your choice");
    scanf("%d", &choice);
    switch(choice){
    case 1:
        printf("Enter the data to be inserted:");
        scanf("%d", &data);
        head = insertNode(head, data);
        break;
    case 2: head = bubbleSort(head);
        printf("linked list sorted \n");
        break;
    case 3: head = reverselist(head);
        printf("linked list reversed \n");
        break;
    case 4: printf("Enter the data for the second
            linked list");
        struct Node * head2 = NULL;
        while(1){
        scanf("%d", &data);
        if(data == -1)
          break;
        head2 = insertNode(head2, data);
      }
      head = concatlist(head, head2);
```

```
printf("Linked lists concatenated\n");
break;
case 5: printlist(head);
        break;
case 6: exit(0);
default: printf("Invalid choice\n");
}
}
return 0;
}
```

Output:

1. InsertNode
2. Sort Linked list
3. Reverse Linked list
4. Concatenate Linked list
5. Print Linked list
6. ~~Enter~~ Exit

Enter your choice: 1
Enter the data to insert: 14

Enter your choice: 1
Enter the data to insert: 10

Enter your choice: 1
Enter the data to insert: 15

Enter your choice: 1
Enter the data to insert: 18

Enter your choice: 2
~~10 14 15~~
List is sorted
Enter your choice: 5
10 14 15 18

Enter your choice: 3
linked list is reversed

Enter your choice: 5
18 15 14 10

Enter your choice: 4
Enter the data for second linked list
12 9 8 7 -1

Enter your choice: 5
18 15 14 10 12 9 8 7

Enter your choice: 6

End
28/11/24

Stack implementation

```
#include <stdio.h>
#include <stdlib.h>
struct Node{
  int data;
  struct Node *next;
};
struct Node *createNode(int data)
{
  struct Node *newNode = (struct Node *)
malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}
void push(struct Node **top, int data)
{
  struct Node *newNode = createNode(data);
  newNode->next = *top;
  *top = newNode;
  printf("%d pushed to the stack\n", data);
}
int pop(struct Node **top) {
  if (*top == NULL)
{
  printf("stack is empty\n");
  return -1;
}
  int poppedvalue = (*top)->data;
  struct Node *temp = *top;
  *top = (*top)->next;
  free(temp);
  return poppedvalue;
}
```

```
void displayStack (struct Node * top)
{
  if (top == NULL)
  {
    printf ("Stack is empty\n");
    return;
  }
  printf ("Stack elements: ");
  while (top != NULL)
    printf ("%d", top -> data);
    top = top -> next;
}
printf ("\n");
}
int main ()
{
  struct Node * stackTop = NULL;
  int choice, data;
  while (1)
  {
    printf ("1. Push\n 2. Pop\n 3. DisplayStack\n
            4. Exit\n");
    printf ("Enter your choice");
    scanf ("%d", &choice);
    switch (choice)
  {
    case 1: printf ("Enter data to push: ");
    scanf ("%d", &data);
    push (&stackTop, data);
    break;
    case 2: printf ("Popped element %d\n",
            pop (&stackTop));
            break;
```

```
case 4: exit(0);
default: printf("Invalid choice");
}
}
return 0;
}
```

Output:

1. Push
2. Pop
3. Display stack
4. Exit

Enter your choice: 1
Enter data to push: 12
12 pushed to the stack

1. Push
2. Pop
3. Display stack
4. Exit

Enter your choice: 1
Enter data to push: 13
13 pushed to the stack

1. Push
2. Pop
3. Display stack
4. Exit

Enter your choice: 1
Enter data to push: 14
14 pushed to the stack

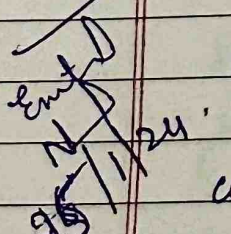
1. Push
2. Pop
3. Display stack
4. Exit
Enter your choice: 1 Enter data to push: 13
13 pushed to the stack
1
1. Push
2. Pop
3. Display stack
4. Exit
Enter your choice: 2
Popped element: 13
1. Push
2. Pop
3. Display stack
4. Exit

Queue implementation

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
  int data;
  struct Node * next;
};
struct node * head = NULL;
void enqueue()
{
  struct node * temp;
  temp = (struct node *) malloc(sizeof(struct
  printf("Enter value to be inserted: \n");
  scanf("%d", &temp -> data);
  temp -> next = NULL;
  if(head == NULL)
  {
    head = temp;
    return;
  }
  else
  {
    struct node * ptr;
    ptr = head;
    while(ptr -> next != NULL)
    {
      ptr = ptr -> next;
    }
    ptr -> next = temp;
  }
}
```

```
void dequeue()
{
  struct node *ptr;
  if (head == NULL)
  {
    printf("Queue is empty\n");
    return;
  }
  else
  {
    ptr = head;
    head = ptr->next;
    printf("value dequeue = %d", ptr->data);
    free(ptr);
  }
}
void display()
{
  if (head == NULL)
  {
    printf("Queue is empty\n");
    return;
  }
  else
  {
    struct node *ptr;
    ptr = head;
    while (ptr != NULL)
    {
      printf("%d\t", ptr->data);
      ptr = ptr->next;
    }
  }
}
```

```
void main ()
{
int choice;
while(1)
{
printf("1.enqueue\n");
printf("2.dequeue\n");
printf("3.display\n");
printf("4.exit\n");
printf("Enter your choice");
scanf("%d",&choice);
switch(choice)
{
case 1: enqueue();
        break;
case 2: dequeue();
        break;
case 3: display();
        break;
case 4: exit(0);
        break;
}
}
}
```



output
1. enqueue
2. dequeue
3. display
4. exit
enter your choice: 1
enter value to be inserted: 12

enter your choice: 1
enter value to be inserted: 54

enter yourchoice: 1
enter value to be inserted: 19

enter your choice: 1
enter value to be inserted: 20

enter your choice: 1
enter value to be inserted: 30

enter your choice: 3
enter value to be inserted: 12 54 19 20 30

enter your choice: 2
Value dequeue 12

enter your choice: 2
value dequeue: 54

enter your choice: 3
19 20 30