Concatenate, sort, reverse :- 25/1/24

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

struct Node
    {
      int data;
      struct Node *next;
    };
void append (struct Node* *head_ref, int new_data)
   {
     struct Node* new_node = (struct Node *) malloc (sizeof
                                       (struct Node));

     struct Node * last = *head_ref;
     new_node->data = new_data;
     new_node->next = NULL;

     if (*head_ref == NULL) {
                 {
                   * head_ref = new_node;
                   return;
                 }
     while (last->next != NULL)
         {
           last = last->next;
         }
     last->next = new_node;
   }
void printList (struct node *node)
         {
         while (node != NULL)
               {
                printf ("%d ->", node->data);
                node = node->next;
               }
         printf ("Null\n");
       }
```

```
void sortList (struct Node** head_ref)
{
if(*head_ref == NULL)
  {
   return;
  }
int swapped, temp;
struct Node *ptr1;
struct Node *lptr = NULL;

do
 {
  swapped = 0;
  ptr1 = *head_ref;
  while (ptr1->next != lptr)
       {
        if(ptr1->data > ptr1->next->data)
          {
           temp = ptr1->data;
           ptr1->data = ptr1->next->data;
           ptr1->next->data = temp;
           swapped = 1;
          }
         ptr1 = ptr1->next;
       }
       lptr = ptr1;
  }
  while(swapped);
}
~~void~~
```

```
void reverseList(struct Node* *head_ref)
  {
    struct Node* prev = NULL;
    struct Node* current = *head_ref;
    struct Node* next = NULL;

    While (current != NULL)
        {
          next = current->next;
          current->next = prev;
          prev = current;
          current = next;
        }
      * head_ref = prev;
  }

void concatenateLists(struct Node* *head1, struct Node* head2)
  {
    if(* head1 == NULL)
        {
          * head1 = head2;
          return;
        }

    struct Node* temp = * head1;
    while (temp -> next != NULL)
        {
          temp = temp->next;

        }
      temp->next = head2;
  }
int main ()
  {
    struct Node *list1 = NULL;
    struct Node *list2 = NULL;
    int n, data;
printf("Enter the number of elements for list 1:");
scanf("%d", &n);

printf("Enter the elements for List 1:\n");
for (int i=0; i<n; i++)
    {
     scanf("%d", &data);
     append(&list1, data);

    }
 printf(" Enter the number of elements for List 2:");
 scanf("%d", &n);
 printf(" Enter the elements for List 2:\n");
 for (int i=0; i<n; i++)
     {
      scanf("%d", &data);
      append(&list2, data);
     }
 printf("\n Original list1:");
 printList(list1);
 printf("Original List 2: ");
 printList(list2);

 sortList(&list1);
 sortList(&list2);
 printf("\nSorted list 1:");
 printList(list1);
 printf("Sorted list 2:");
 printList(list2);          26/11/24

 concatenateLists(&list1, list2);
 printf("\n concatenated list:");
 printList(list1);
 reverseList(&list1);
 printf("\n Reversed List:");
 printList(list1);
 return 0;
}
```

2) Stack using Linked List

```
#include <stdio.h>
#include <stdlib.h>
void push();
void pop();
void display();
struct node
   {
     int val;
     struct node * next;
   };
struct node *head;
void main()
 {
     int choice = 0;
     printf("\n stack operation using linked list \n");
     while (choice != 4)
        {
          printf(" \n choose one from the below
                          options... \n");
          printf("\n1. Push \n2. pop \n3. show \n4. exit");
          printf("\n Enter your choice \n");
          scanf("%d", &choice);
          switch (choice)
          {
             case 1 :
               {
                 push();
                  break;
               }
             Case 2 :
               {
                 pop();
                  break;
               }
```

```
Case 3 :
{
    display( );
    break ;
  }
  Case 4 :
  {
    printf ("Exiting..");
    break ;

  }
  default :
  {
    printf (" Please enter valid choice");
  }

 };
}
void push ()
{
  int val ;
  struct node *ptr = (struct node *) malloc(sizeof (struct node));
  if (ptr == NULL)
  {
   printf("not able to push the element ");
  }
  else
  { printf(" Enter the value ");
   scanf(" %d" , &val );
   if (head == Null)
     {
       ptr->val =val;
       ptr ->next =NULL;
       head = ptr ;
     }
   else
   {
      ptr->val = val;
      ptr-> next = head ;
      head = ptr ;
   }
   printf(" Item pushed");
```

[illegible] 25/1/24

```
void pop()
{
    int item;
    struct node *ptr;
    if (head == NULL)
      {
        printf("underflow");
      }
    else
      {
        item = head->val;
        ptr = head;
        head = head->next;
        free(ptr);
        printf(" Item popped ");
      }
}
void display()
 {
    int i;
    struct node *ptr;
    ptr = head;
    if(ptr == NULL)
      {
        printf("stack is empty\n");
      }
    else
      {
        printf("Printing stack elements \n");
        while(ptr != NULL)
          {
            printf("%d \n", ptr->val);
            ptr = ptr->next;
          }
      }
 }
```

3) Queue using Single linked list :

```
#include <stdio.h>
#include<stdlib.h>
struct node
{
  int data;
  struct node *next;
};
struct node *front;
struct node *rear;
void insert ();
void delete ();
void display();
void main ()
{
    int choice;
    while (choice !=4)
     {
       printf (" \n  Queue operation using linked list ");
       printf(" \n 1. insert an element \n 2. Delete an element
              \n 3. Display the queue \n 4. exit \n");
       printf (" \n Enter your choice ");
       scanf ("%d" , &choice);
       switch(choice)
        {
          case 1 :
          insert();
          break;
         case 2 :
           delete ();
           break;
         case 3:
           display();
           break;
         case 4:
           exit(0);
           break;
        default:
           printf("\nEnter valid choice\n");
        }
}  }
```

[illegible]
25/1/24

```
void insert()
 {
   struct node *ptr;
   int item;
   ptr = (struct node *) malloc (sizeof(struct node));
   if(ptr == NULL)
      {
        printf("\n Overflow \n");
        return;
      }
   else
     {
       printf("\n Enter value \n");
       scanf("%d", &item);
       ptr->data = item;
       if(front == NULL)
         {
            front = ptr;
            rear = ptr;
            front->next = NULL;
            rear->next = NULL;
         }
       else
         {
            rear->next = ptr;
            rear = ptr;
            rear->next = NULL;
         }
     }
 }
void delete()
 {
   struct node *ptr;
   if(front == NULL)
     {
       printf("\n underflow \n");
       return;
     }
   else
     {
       ptr = front;
       front = front->next;
       free(ptr);
     }
 }
```

```
void display()
{
 struct node *ptr;
 ptr = front;
 if(front == NULL)
   {
     printf("\nEmpty queue\n");
   }
  else
   {
     printf("\n printing values..\n");
     while (ptr != NULL)
       {
         ~~ptr~~ printf("\n %d \n", ptr->data);
             ptr = ptr->next;
       }
   }
}
```

o/p:

```
~~Enter the choice~~
Queue operation using linked list
    1. insert an element
    2. delete an element
    3. display the queue
    4. exit
   enter the choice   1
   enter the value
      2
    enter the choice  1
    enter the value
      66
     enter the choice 3
      printing values
        2
        66
     enter choice 2
     enter the choice 3
        66
```