1. SLL - sort, reverse, concatenation

```
struct node *sort (struct node *head)
{
    struct node *ptr1, *ptr2;
    int temp;
    ptr1 = head;
    while (ptr1->next != NULL)
    {
        ptr2 = ptr1->next;
        while (ptr2 != NULL)
        {
            if (ptr1->data > ptr2->data)
            {
                temp = ptr1->data;
                ptr1->data = ptr2->data;
                ptr2->data = temp;
            }
            ptr2 = ptr2->next
        }
        ptr1 = ptr1->next
```

```
struct node * reverse (struct node *head)
{
    struct node *ptr, *prev;
    while (head != NULL)
    {
        ptr = head->next;
        head->next = prev;
        prev = head;
        head = ptr;
    }
    head prev;
    display(head);
}
```

```
struct node
{
    int data;
    struct node *next;
};

struct node *head1 = NULL;
struct node *head2 = NULL;

struct node *insert(struct node *head)
{
    struct node *new_node, *ptr;
    printf("Enter the number");
    scanf("%d", &num);
    new_node = (struct node*) malloc(sizeof(struct node));
    new_node->data = num;
    new_node->next = NULL;
    if (head == NULL)
    {
        head = new_node;
    }
    else
    {
        ptr = head;
        while (ptr->next != NULL)
        {
            ptr = ptr->next;
        }
        ptr->next = new_node;
    }
    return head;
}

void concat(struct node *head1, struct node *head2)
{
    struct node *ptr;
    if (head1 != NULL && head2 != NULL)
    {
        ptr = head1;
        while (ptr->next != NULL
        {
            ptr = ptr->next;
```

```
        ptr -> next = head;
    }
    else
    {
        printf("either of the linked list is empty \n");
    }
    display(head1);
}
```

O/P:

1.insert 1. 2. interest2 . 3. Display1 4. display2 5. concat 6. Sort . 7. reverse.

Enter your choice : 1

enter number: 32

enter choice : 1

enter number : 32

enter choice : 1

enter number : 65

enter ter choice : 1

enter number : 94

enter your choice : 2

enter number : 65

enter choice : 2

enter number : 95

enter choice : 2

enter number : 65

enter choice : 2

enter number : 85

enter choice : 2

enter number : 87

enter choice : 2

enter number : 69

enter choice : 3

elements inside list 1 : 32 → 32 → 65 → 94

enter choice : 4

elements inside list 2 : 65 → 85 → 87 → 69.

enter choice : 5

elements in linked list : 32 -> 32 -> 65 -> 94 -> 65 -> 85 -> 87 -> 69

enter choice : 6

elements in linked list : 32 -> 32 -> 65 -> 65 -> 69 -> 85 -> 87 -> 94

enter choice : 7

elements in linked list : 94 -> 87 -> 85 -> 69 -> 65 -> 65 -> 32 -> 32

2. Stack implementation using single linked list.

```
struct node
{
   int data;
   struct node *next;
};
struct node *head = NULL;
void push()
{
   struct node *new_node;
   int num;
   printf("Enter number: ");
   scanf("%d", &num);
   new_node = (struct node *)malloc(sizeof(struct node));
   new_node->data = num;
   if (head == NULL)
   {
      head = new_node;
      new_node->next = NULL;
   }
   else
   {
      new_node->next = head;
      head = new_node;
   }
}
void pop()
{
   struct node *ptr;
```

```
    if (head = NULL)
    {
      printf("underflow \n");
    }
    else
    {
      ptr = head;
      head = ptr->next;
      printf("poped element: %d \n", ptr->data);
      free(ptr);
    }
}

void display()
{
  struct node *ptr;
  ptr = head;
  printf("elements of linked list\n");
  while (ptr != NULL)
  {
    printf("%d ->", ptr->data);
    ptr = ptr->next;
  }
  printf
}

int main()
{
  int ch;
  while (1)
  {
    printf("Enter the choice");
    scanf("%d", &ch);
    switch (ch)
    {
      case 1:
        push();
        break;
      case 2:
        pop();
        break;
      case 3:
        display()
        break;
      case 4: exit(0);
```

```
        default:
            printf("invalid number");
        }
    }
}
```

o/p:

enter choice: 1

enter number: 32

enter choice: 1

enter number: 35

enter choice: 1

enter number: 65

enter choice: 3.

elements of linked list: 65 -> 35 -> 32

enter choice = 2

poped element: 65

~~elect~~ enter choice: 3

elements of linked list 35 -> 32.

.

Jaw

5/2/24

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

struct node
{
    int data;
    struct node *next;
};

struct node *head1;
struct node *head2;

struct node *insert(struct node *head)
{
    struct node *new_node, *ptr;
    int num;

    printf("Enter the number: ");
    scanf("%d", &num);

    new_node = (struct node *)malloc(sizeof(struct node));
    new_node->next = NULL;
    new_node->data = num;

    if (head == NULL)
    {
        head = new_node;
    }
    else
    {
        ptr = head;
        while (ptr->next != NULL)
        {
            ptr = ptr->next;
        }
        ptr->next = new_node;
    }
```

```
38      return head;
39  }
40
41  void display(struct node *head)
42  {
43      struct node *ptr;
44
45      ptr = head;
46      printf("Element are: \n");
47      while (ptr != NULL)
48      {
49          printf("%d->", ptr->data);
50          ptr = ptr->next;
51      }
52      printf("\n");
53  }
54
55  void concat(struct node *head1, struct node *head2)
56  {
57      struct node *ptr;
58
59      if (head1 != NULL && head2 != NULL)
60      {
61          ptr = head1;
62          while (ptr->next != NULL)
63          {
64              ptr = ptr->next;
65          }
66          ptr->next = head2;
67          head2=head1;
68      }
69      else
70      {
71          printf("either of the Linked List is Empty");
72      }
73      display(head1);
```

```
    display(head1);
}

void reverse(struct node *head1)
{
    struct node *ptr = NULL, *prev = NULL;
    while (head1 != NULL)
    {
        ptr = head1->next;
        head1->next = prev;
        prev = head1;
        head1 = ptr;
    }
    head1 = prev;
    display(head1);
}

void sort(struct node *head1)
{
    struct node *ptr1,*ptr2;
    int temp;
    ptr1=head1;
    while(ptr1->next!=NULL)
    {
            ptr2=ptr1->next;
            while(ptr2!=NULL)
            {
                if(ptr1->data>ptr2->data)
                {
                    temp=ptr1->data;
                    ptr1->data=ptr2->data;
                    ptr2->data=temp;
                }
                ptr2=ptr2->next;
            }
            ptr1=ptr1->next;
    }

}

```

```c
int main()
{
    int choice;

    while (1)
    {
        printf("1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse\n");
            printf("Enter the Choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            head1 = insert(head1);
            break;
        case 2:
            head2 = insert(head2);
            break;
        case 3:
            display(head1);
            break;
        case 4:
            display(head2);
            break;
        case 5:
            concat(head1, head2);
            break;
        case 6:
            sort(head1);
            break;
        case 7:
            reverse(head1);
            break;
        }
    }
}
```

```
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 1
Enter the number: 32
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 1
Enter the number: 32
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 1
Enter the number: 65
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 1
Enter the number: 94
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 2
Enter the number: 65
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 2
Enter the number: 95
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 2
Enter the number: 65
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 2
Enter the number: 85
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 2
Enter the number: 87
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 2
Enter the number: 69
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 3
Element are:
32->32->65->94->
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 4
Element are:
65->95->65->85->87->69->
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 5
```

```
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 2
Enter the number: 87
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 2
Enter the number: 69
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 3
Element are:
32->32->65->94->
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 4
Element are:
65->95->65->85->87->69->
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 5
Element are:
32->32->65->94->65->95->65->85->87->69->
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 6
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 3
Element are:
32->32->65->65->65->69->85->87->94->95->
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: 7
Element are:
95->94->87->85->69->65->65->65->32->32->
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice:
```

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

struct node
{
    int data;
    struct node *next;
};

struct node *head = NULL;
void push()
{
    struct node *new_node;
    int num;
    printf("Enter the number:\n");
    scanf("%d", &num);
    new_node = (struct node *)malloc(sizeof(struct node));
    new_node->data = num;
    if (head == NULL)
    {
        head = new_node;
        new_node->next = NULL;
    }
    else
    {
        new_node->next = head;
        head = new_node;
    }
}

void pop()
{
    struct node *ptr;
    if (head == NULL)
    {
        printf("underflow\n");
    }
    else
    {
        ptr = head;
        head = ptr->next;
        printf("popped element:%d", ptr->data);
        free(ptr);
    }
}
```

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

int main()
{
    int ch;
    while (1)
    {
        printf("-------menu------\n1.push\n2.pop\n3.display\n4.exit\n");
        printf("Enter the choice\n");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            push();
            break;
        case 2:
            pop();
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
            break;
        default:
            printf("invalid number!");
        }
    }
}
```

```
-------menu------
1.push
2.pop
3.display
4.exit
Enter the choice
1
Enter the number:
32
-------menu------
1.push
2.pop
3.display
4.exit
Enter the choice
1
Enter the number:
35
-------menu------
1.push
2.pop
3.display
4.exit
Enter the choice
1
Enter the number:
65
-------menu------
1.push
2.pop
3.display
4.exit
Enter the choice
3
65
35
32
```

```
-------menu------
1.push
2.pop
3.display
4.exit
Enter the choice
2
popped element:65-------menu------
1.push
2.pop
3.display
4.exit
Enter the choice
3
35
32
-------menu------
1.push
2.pop
3.display
4.exit
Enter the choice
```