

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 connect(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 = head2;
    }
}

```

ptr → next 2 head;

{
else

{

printf ("either of the linked list is empty (%d)\n");
}

display (head1);

}

0/1:

1.insert 2.interate 3.Display 4.display2 Scraml 6.Sort 7,reverse.

1.

enter number: 32

enter choice: 1

enter number: 32

enter choice: 1

enter number: 65

enter choice: 1

enter number: 94

enter your choice: 2

enter number: 65

enter choice: 2

enter number: 95

another 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 list1: 32 → 32 → 65 → 94

enter choice: 4

elements inside list2: 65 → 85 → 87 → 69.

enter choice : 5

elements in linked list : 32 → 32 → 63 → 63 → 69 → 85 → 87 → 94

enter choice : 6

elements in linked list : 32 → 32 → 63 → 63 → 69 → 85 → 87 → 94

enter choice : 7

elements in linked list : 94 → 87 → 85 → 69 → 63 → 63 → 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 *t;

```

if (head == NULL)
{
    printf("Underflow (m),");
}
else
{
    pptr = head;
    head = pptr->next;
    printf("Popped element: (%d,%d,%d,%d), pptr->data");
    free(pptr);
}
}

```

void display()

{

struct nod *pptr;

pptr = head;

printf("Elements of linked list:");

while (pptr != NULL)

{

printf("%d %d", pptr->data);

pptr = pptr->next;

}

~~printf~~

}

int main()

{

int ch;

while (1)

printf("Enter the choice");

scanf("(l,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: 33

enter choice: 1

enter number: 65

enter choice: 3.

elements of linked list: 65 → 33 → 32

enter choice = 2

popped element: 65

~~enter choice~~

elements of linked list 33 → 32.

for
5/2/2021

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct node
5 {
6     int data;
7     struct node *next;
8 };
9
10 struct node *head1;
11 struct node *head2;
12
13 struct node *insert(struct node *head)
14 {
15     struct node *new_node, *ptr;
16     int num;
17
18     printf("Enter the number: ");
19     scanf("%d", &num);
20
21     new_node = (struct node *)malloc(sizeof(struct node));
22     new_node->next = NULL;
23     new_node->data = num;
24
25     if (head == NULL)
26     {
27         head = new_node;
28     }
29     else
30     {
31         ptr = head;
32         while (ptr->next != NULL)
33         {
34             ptr = ptr->next;
35         }
36         ptr->next = new_node;
37     }
}
```

```
38     return head;
39 }
40
41 void display(struct node *head)
42 {
43     struct node *ptr;           I
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);
```

```
73     display(head1);
74 }
75
76 void reverse(struct node *head1)
77 {
78     struct node *ptr = NULL, *prev = NULL;
79     while (head1 != NULL)
80     {
81         ptr = head1->next;
82         head1->next = prev;
83         prev = head1;
84         head1 = ptr;
85     }
86     head1 = prev;
87     display(head1);
88 }
89
90 void sort(struct node *head1)
91 {
92     struct node *ptr1,*ptr2;
93     int temp;
94     ptr1=head1;
95     while(ptr1->next!=NULL)
96     {
97         ptr2=ptr1->next;
98         while(ptr2!=NULL)
99         {
100             if(ptr1->data>ptr2->data)
101             {
102                 temp=ptr1->data;
103                 ptr1->data=ptr2->data;
104                 ptr2->data=temp;
105             }
106             ptr2=ptr2->next;
107         }
108         ptr1=ptr1->next;
109     }
110 }
111 }
```

```
112
113 int main()
114 {
115     int choice;
116
117     while (1)
118     {
119         printf("1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse\n");
120         printf("Enter the Choice: ");
121         scanf("%d", &choice);
122
123         switch (choice)
124         {
125             case 1:
126                 head1 = insert(head1);
127                 break;
128             case 2:
129                 head2 = insert(head2);
130                 break;
131             case 3:
132                 display(head1);
133                 break;
134             case 4:
135                 display(head2);
136                 break;
137             case 5:
138                 concat(head1, head2);
139                 break;
140             case 6:
141                 sort(head1);
142                 break;
143             case 7:
144                 reverse(head1);
145                 break;
146         }
147     }
148 }
149 }
```

```
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->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->32->32->
1.insert1 2.insert2 3.display1 4.display2 5.concat 6.sort 7.reverse
Enter the Choice: |

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct node
5 {
6     int data;
7     struct node *next;
8 };
9
10 struct node *head = NULL;
11 void push()
12 {
13     struct node *new_node;
14     int num;
15     printf("Enter the number:\n");
16     scanf("%d", &num);
17     new_node = (struct node *)malloc(sizeof(struct node));
18     new_node->data = num;
19     if (head == NULL)
20     {
21         head = new_node;
22         new_node->next = NULL;
23     }
24     else
25     {
26         new_node->next = head;
27         head = new_node;
28     }
29 }
30
31 void pop()
32 {
33     struct node *ptr;
34     if (head == NULL)
35     {
36         printf("underflow\n");
37     }
38     else
39     {
40         ptr = head;
41         head = ptr->next;
42         printf("popped element:%d", ptr->data);
43         free(ptr);
44     }
45 }
```

```
47 void display()
48 {
49     struct node *ptr;
50     ptr = head;
51     if (head == NULL)
52     {
53         printf("stack is empty\n");
54     }
55     else
56     {
57         while (ptr != NULL)
58         {
59             printf("%d\n", ptr->data);
60             ptr = ptr->next;
61         }
62     }
63 }
64
65 int main()
66 {
67     int ch;
68     while (1) I
69     {
70         printf("-----menu-----\n1.push\n2.pop\n3.display\n4.exit\n");
71         printf("Enter the choice\n");
72         scanf("%d", &ch);
73         switch (ch)
74         {
75             case 1:
76                 push();
77                 break;
78             case 2:
79                 pop();
80                 break;
81             case 3:
82                 display();
83                 break;
84             case 4:
85                 exit(0);
86                 break;
87             default:
88                 printf("invalid number!");
89         }
90     }
91 }
```

-----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