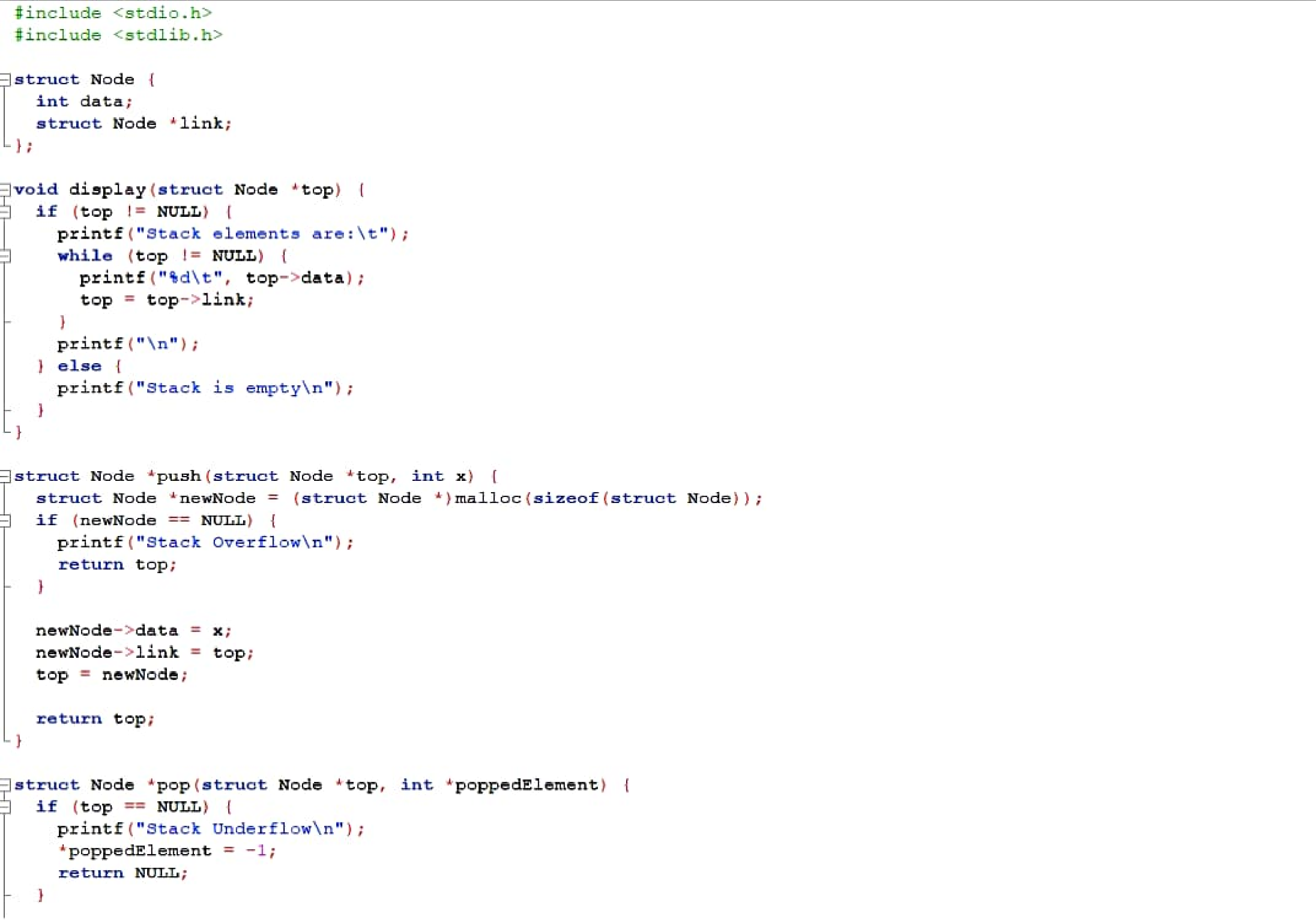
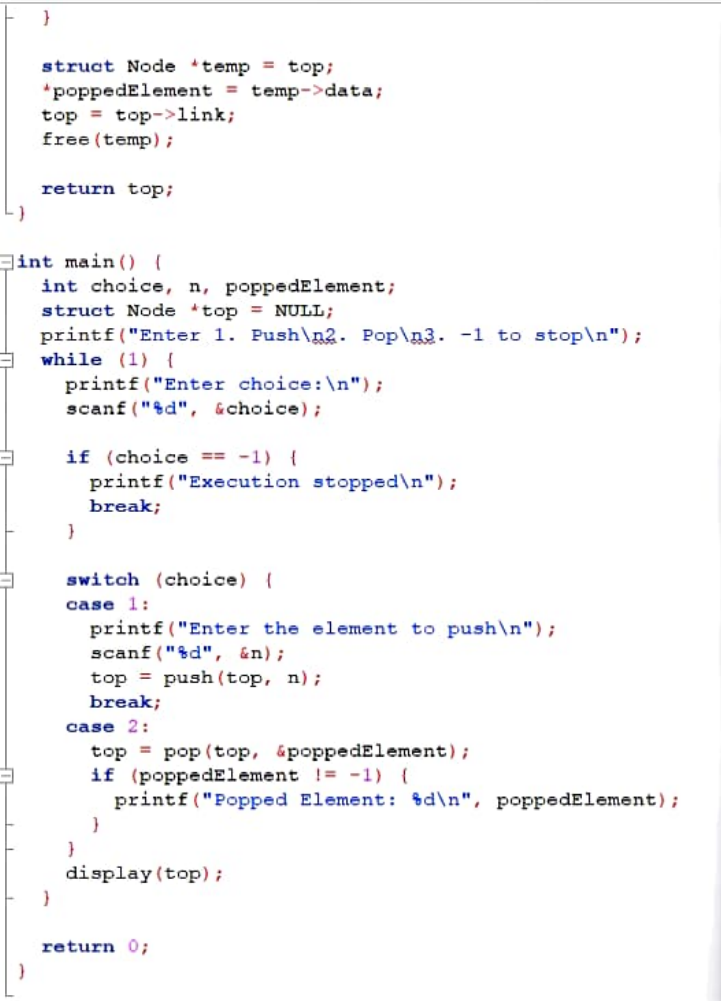
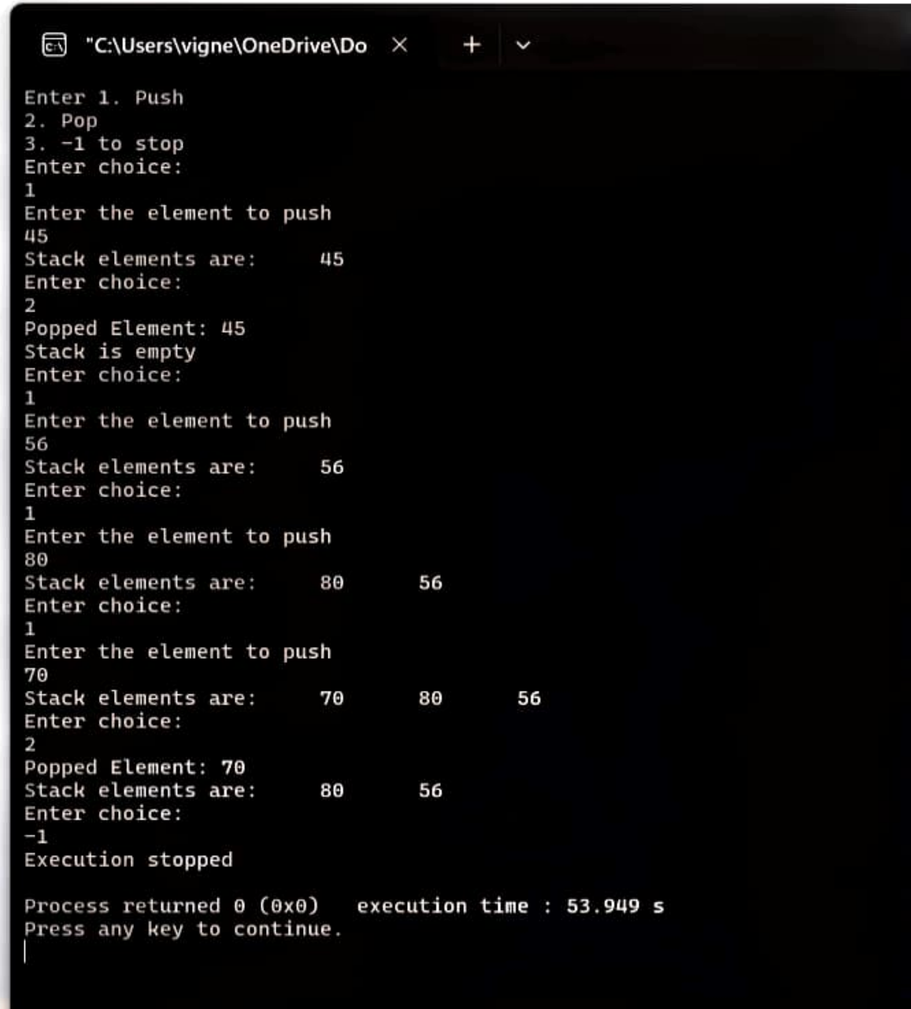


```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
  int data;
  struct Node *link;
};

void display(struct Node *top) {
  if (top != NULL) {
    printf("Stack elements are:\t");
    while (top != NULL) {
      printf("%d\t", top->data);
      top = top->link;
    }
    printf("\n");
  } else {
    printf("Stack is empty\n");
  }
}

struct Node *push(struct Node *top, int x) {
  struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
  if (newNode == NULL) {
    printf("Stack Overflow\n");
    return top;
  }

  newNode->data = x;
  newNode->link = top;
  top = newNode;

  return top;
}

struct Node *pop(struct Node *top, int *poppedElement) {
  if (top == NULL) {
    printf("Stack Underflow\n");
    *poppedElement = -1;
    return NULL;
  }
```



```
  }

  struct Node *temp = top;
  *poppedElement = temp->data;
  top = top->link;
  free(temp);

  return top;
}

int main() {
  int choice, n, poppedElement;
  struct Node *top = NULL;
  printf("Enter 1. Push\n2. Pop\n3. -1 to stop\n");
  while (1) {
    printf("Enter choice:\n");
    scanf("%d", &choice);

    if (choice == -1) {
      printf("Execution stopped\n");
      break;
    }

    switch (choice) {
    case 1:
      printf("Enter the element to push\n");
      scanf("%d", &n);
      top = push(top, n);
      break;
    case 2:
      top = pop(top, &poppedElement);
      if (poppedElement != -1) {
        printf("Popped Element: %d\n", poppedElement);
      }
    }
    display(top);
  }

  return 0;
}
```



```
Enter 1. Push
2. Pop
3. -1 to stop
Enter choice:
1
Enter the element to push
45
Stack elements are:     45
Enter choice:
2
Popped Element: 45
Stack is empty
Enter choice:
1
Enter the element to push
56
Stack elements are:     56
Enter choice:
1
Enter the element to push
80
Stack elements are:     80      56
Enter choice:
1
Enter the element to push
70
Stack elements are:     70      80      56
Enter choice:
2
Popped Element: 70
Stack elements are:     80      56
Enter choice:
-1
Execution stopped

Process returned 0 (0x0)   execution time : 53.949 s
Press any key to continue.
```

29/01/24

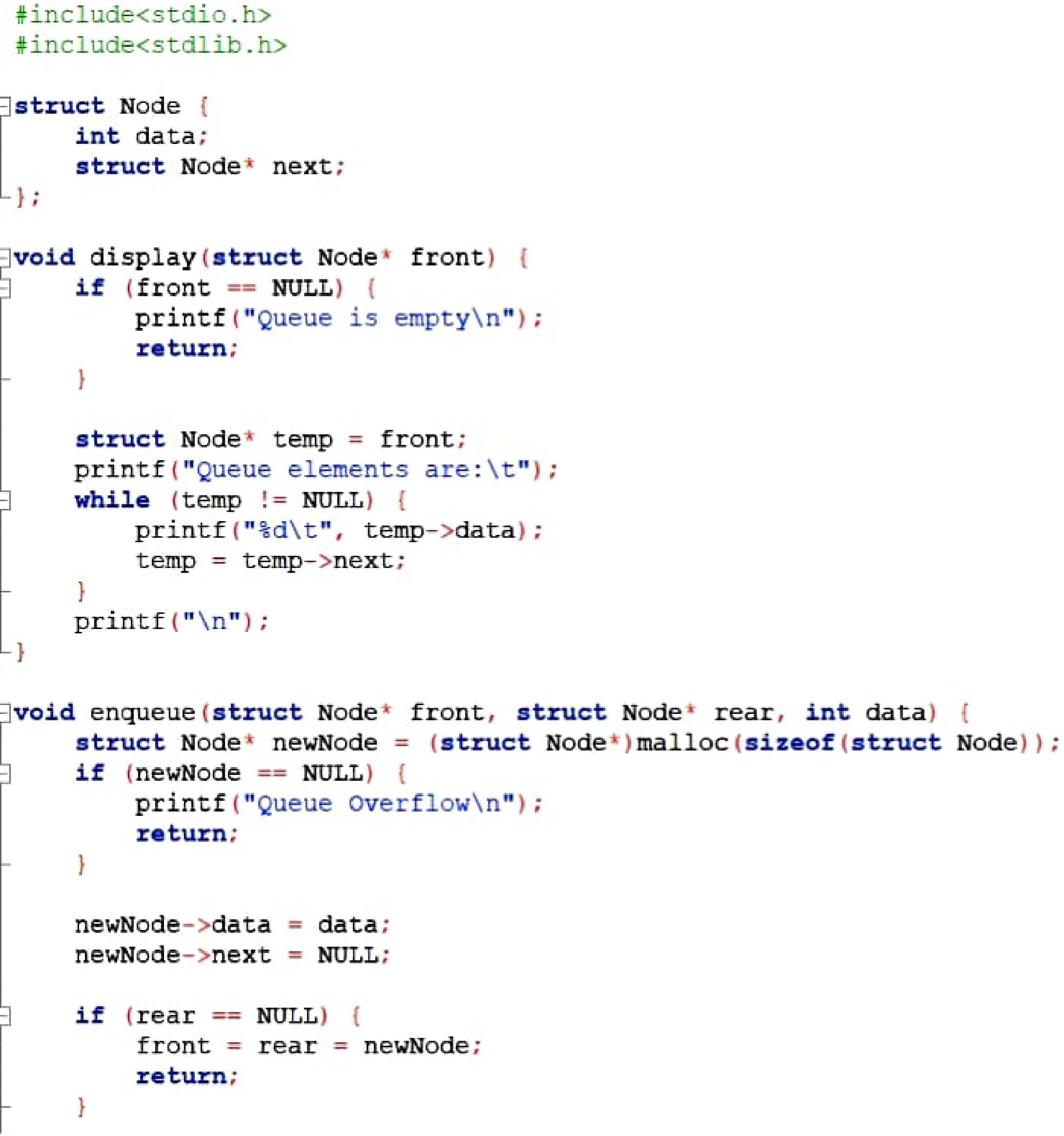
20. Stack implementation using single linked list

```
struct node
{
  int data;
  struct node * link;
}
struct node * top = 0;
void push (int x)
{
 struct node * newnode;
 newnode = (struct node *) malloc(size of (structnode));
  newnode → data = x;
  newnode → link = top;
      top = newnode
 }
void display()
 {
  struct node * temp;
   temp = top;
  if (top==0)
   {
     printf("empty");
   }
 else
 {
  While (temp! =0)
   {
     printf("%d", temp → data);
     temp = temp → link;
   }
 }
```

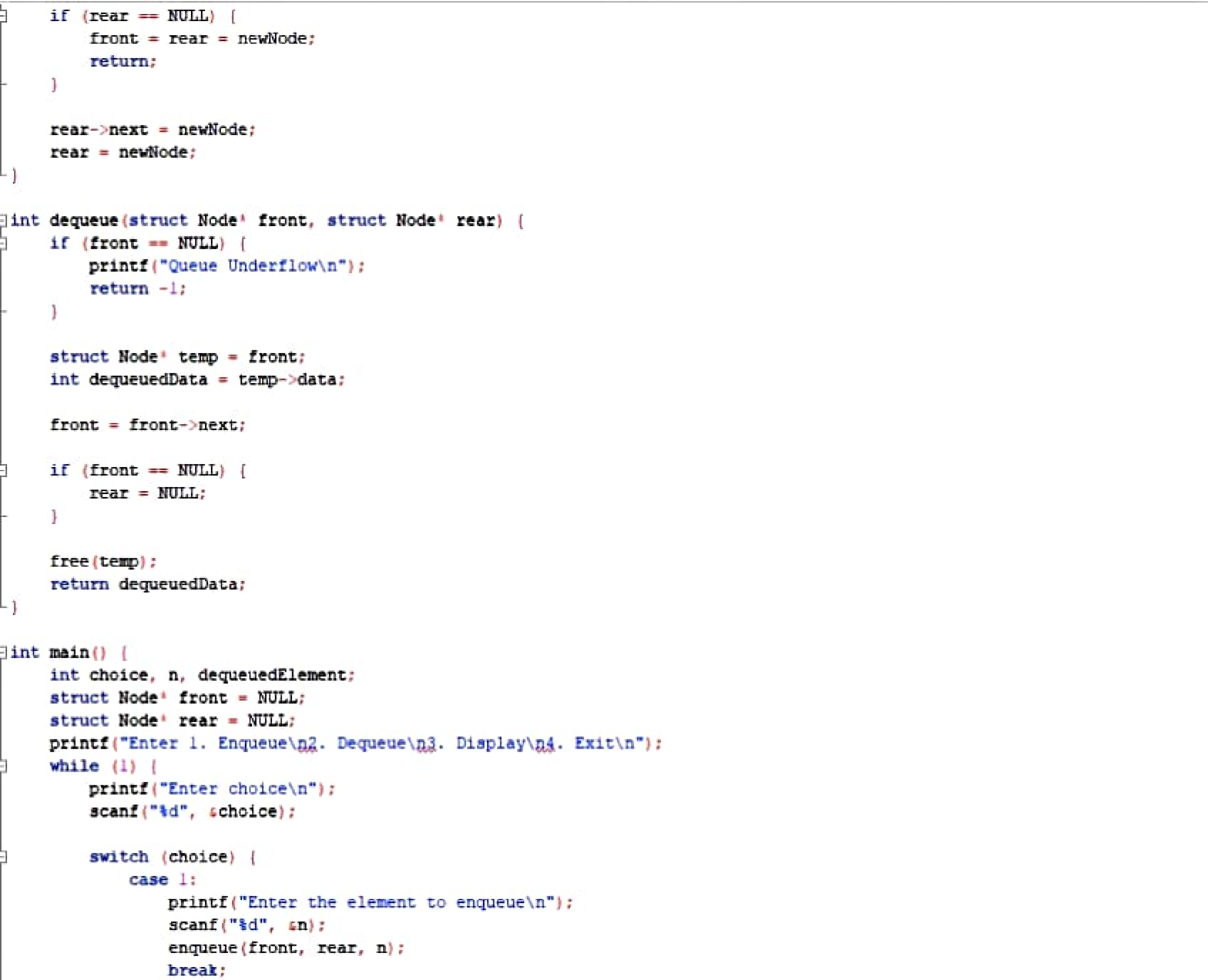
```
void pop()
 {     temp=top;
   ~~struct~~ node *temp;
    if (top == 0)
   {
     printf(" Empty");
     }
     else
     {
       printf("%d", top→data)
       top = top →link;
         free(temp);
       }
     }
```

Output

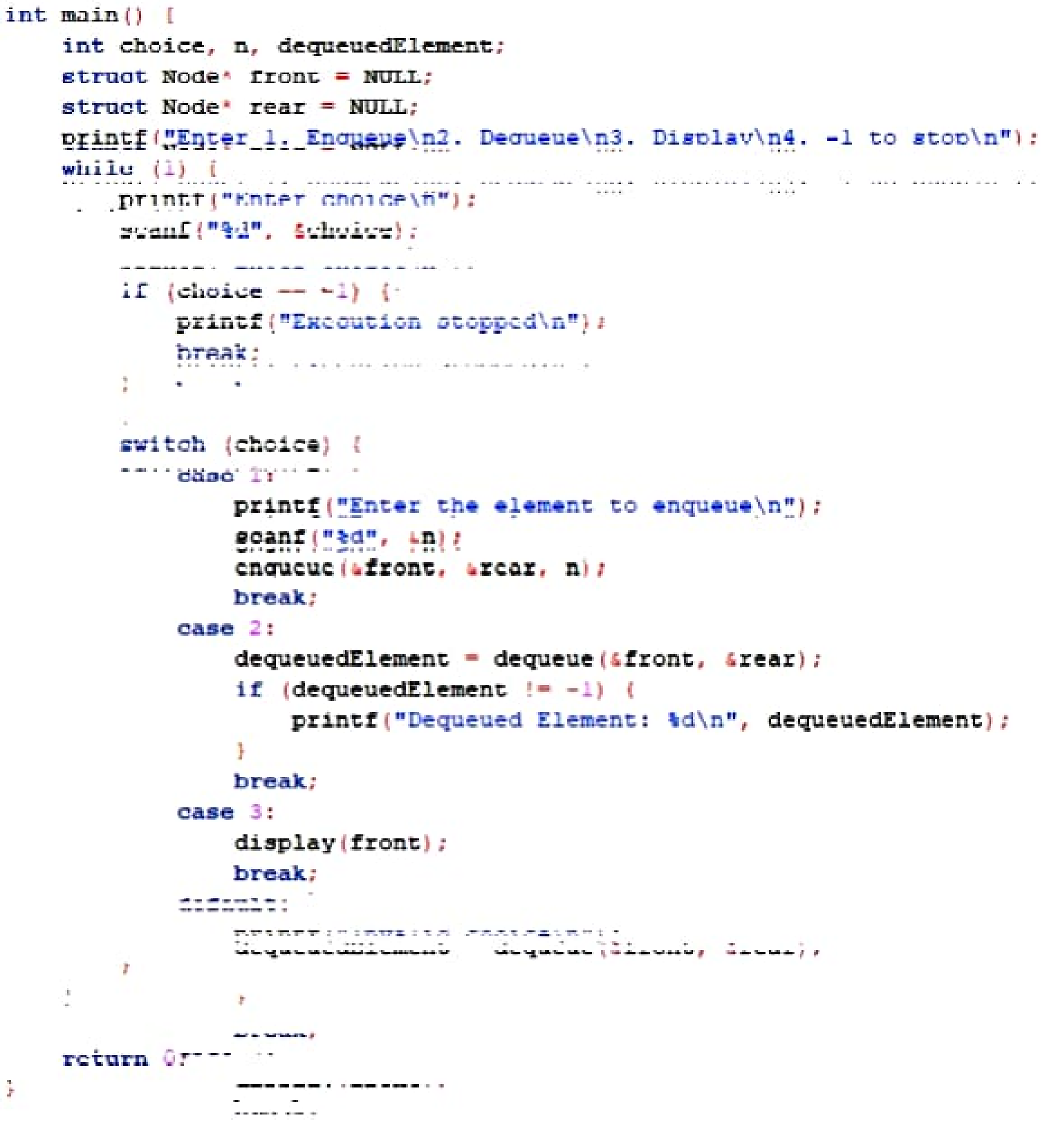
```
Enter choice    1. Push   2. Pop   3. Display   4. Exit
1
Enter the element  5
Enter  choice  1. Push   2. Pop  3. Display   4 Exit
1
Enter  the element  10
Enter  choice  1. Push   2. Pop   3. Display  4. Exit
1
Enter  the element  15.
Enter  choice   1. Push   2. Pop   3. Display   4 Exit.
1
Enter  the element 20
Enter  choice   1. Push   2. Pop   3. Display   4 Exit
3
~~15 10~~  20   15   10   5
Enter  choice   1. Push   2. Pop   3. Display   4. Exit
2
Enter  choice   1. Push   2. Pop   3. Display   4. Exit.
2
Enter     choice   1. Push  2. Pop   3. Display   4. Exit
3
10    5
Enter    choice   1. Push  2. Pop   3. Display   4. Exit
4
```



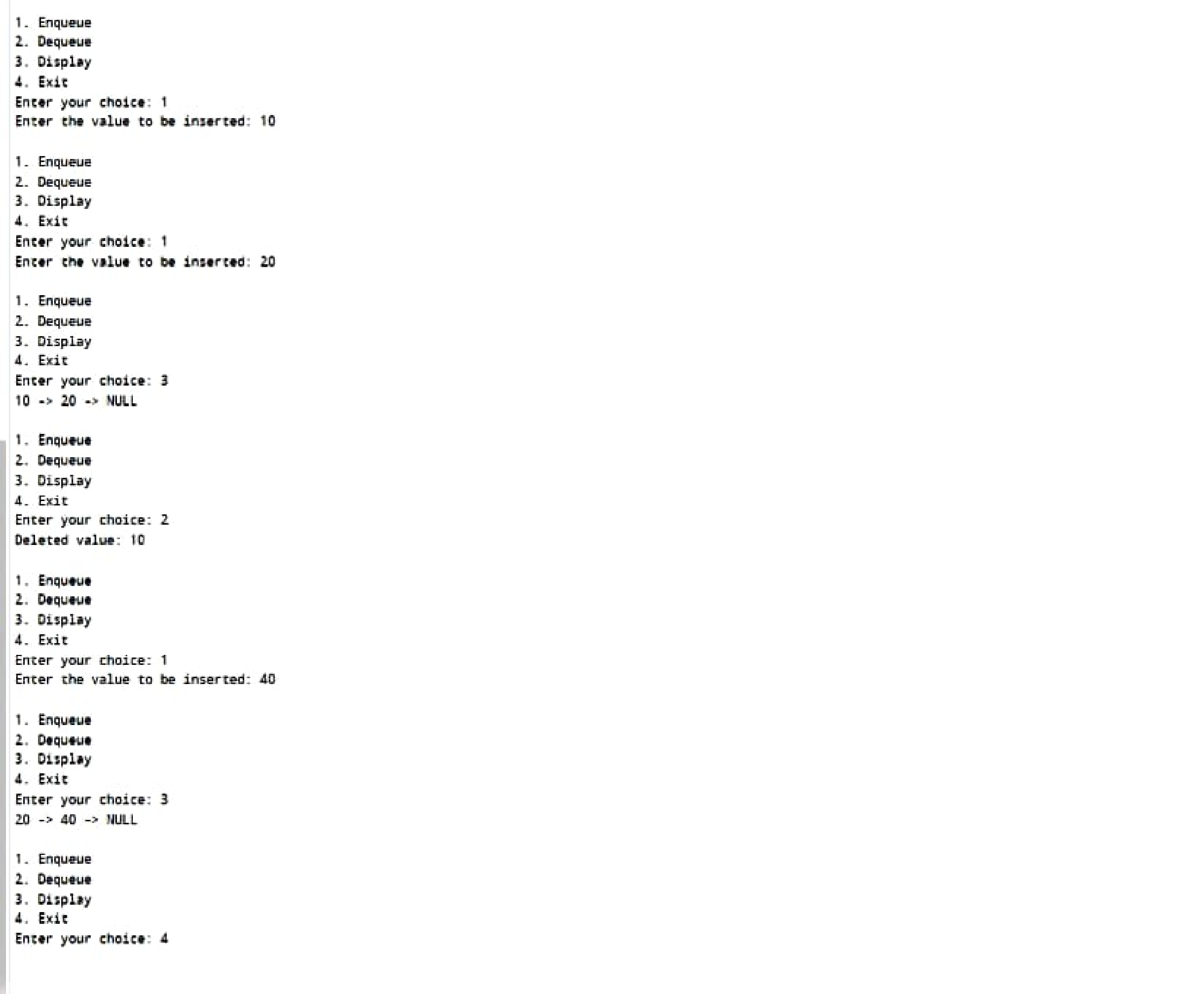
```c
#include<stdio.h>
#include<stdlib.h>

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

void display(struct Node* front) {
    if (front == NULL) {
        printf("Queue is empty\n");
        return;
    }

    struct Node* temp = front;
    printf("Queue elements are:\t");
    while (temp != NULL) {
        printf("%d\t", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

void enqueue(struct Node* front, struct Node* rear, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Queue Overflow\n");
        return;
    }

    newNode->data = data;
    newNode->next = NULL;

    if (rear == NULL) {
        front = rear = newNode;
        return;
    }
```



```
    if (rear == NULL) {
        front = rear = newNode;
        return;
    }

    rear->next = newNode;
    rear = newNode;
}

int dequeue(struct Node* front, struct Node* rear) {
    if (front == NULL) {
        printf("Queue Underflow\n");
        return -1;
    }

    struct Node* temp = front;
    int dequeuedData = temp->data;

    front = front->next;

    if (front == NULL) {
        rear = NULL;
    }

    free(temp);
    return dequeuedData;
}

int main() {
    int choice, n, dequeuedElement;
    struct Node* front = NULL;
    struct Node* rear = NULL;
    printf("Enter 1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\n");
    while (1) {
        printf("Enter choice\n");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the element to enqueue\n");
                scanf("%d", &n);
                enqueue(front, rear, n);
                break;
```



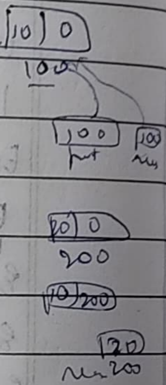
```
int main() {
    int choice, n, dequeuedElement;
    struct Node* front = NULL;
    struct Node* rear = NULL;
    printf("Enter 1. Enqueue\n2. Dequeue\n3. Display\n4. -1 to stop\n");
    while (1) {
        printf("Enter choice\n");
        scanf("%d", &choice);

        if (choice == -1) {
            printf("Execution stopped\n");
            break;
        }

        switch (choice) {
            case 1:
                printf("Enter the element to enqueue\n");
                scanf("%d", &n);
                enqueue(&front, &rear, n);
                break;
            case 2:
                dequeuedElement = dequeue(&front, &rear);
                if (dequeuedElement != -1) {
                    printf("Dequeued Element: %d\n", dequeuedElement);
                }
                break;
            case 3:
                display(front);
                break;
            default:
                printf("Invalid choice\n");
                dequeuedElement = dequeue(&front, &rear);
        }
    }
                break;
    return 0;
}
```



```
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the value to be inserted: 10

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the value to be inserted: 20

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
10 -> 20 -> NULL

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
Deleted value: 10

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the value to be inserted: 40

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
20 -> 40 -> NULL

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 4
```

26 Queue implementation using linked list

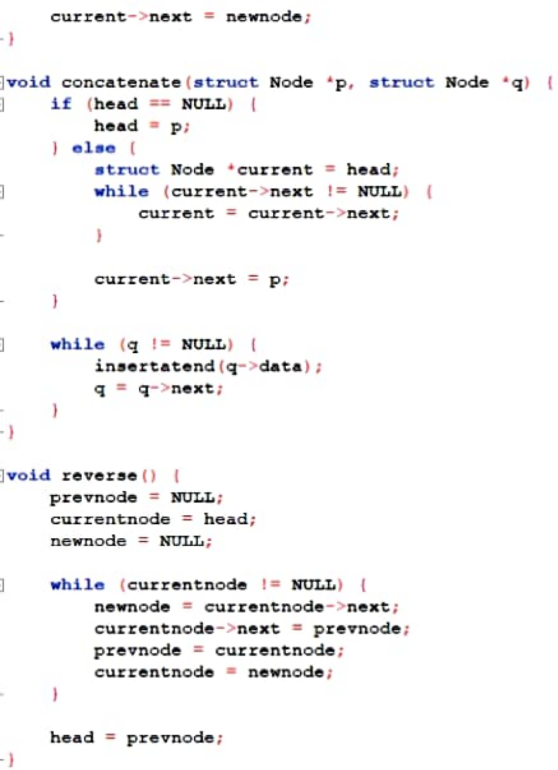
```
struct node{
      int data;
    struct node * next;
   };
struct node * front =0;
struct node * rear =0;

void enqueue(int x)
 {
   struct node * newnode;
   newnode = (structnode *)malloc(sizeof(struct node));
   newnode->data =x;
   newnode->next =0;

    if (front==0 && rear==0)
    {
        front= rear= newnode;
      }
    else
     {
         rear->next = newnode;
         rear = newnode;
      }
  }

void display()
 {
   struct node * temp;
   if (~~fromt~~ ==0 && rear==0)
    {
```



```
printf("Queue is empty");
}
else
    temp = front;
    while (temp!=0)
      {
        printf("%d", temp→data);
        temp = temp→next;
      }
   }

void dequeue()
{
    struct node *temp;

    if (front==0 && rear==0)
     {
        printf("Queue is empty");
     }
    else
       temp = front;
        while (temp != rear)
         {
            ~~temp = temp→next;~~
            ~~free(temp)~~
            front = front→next;
            free(temp);
         }
       }
```

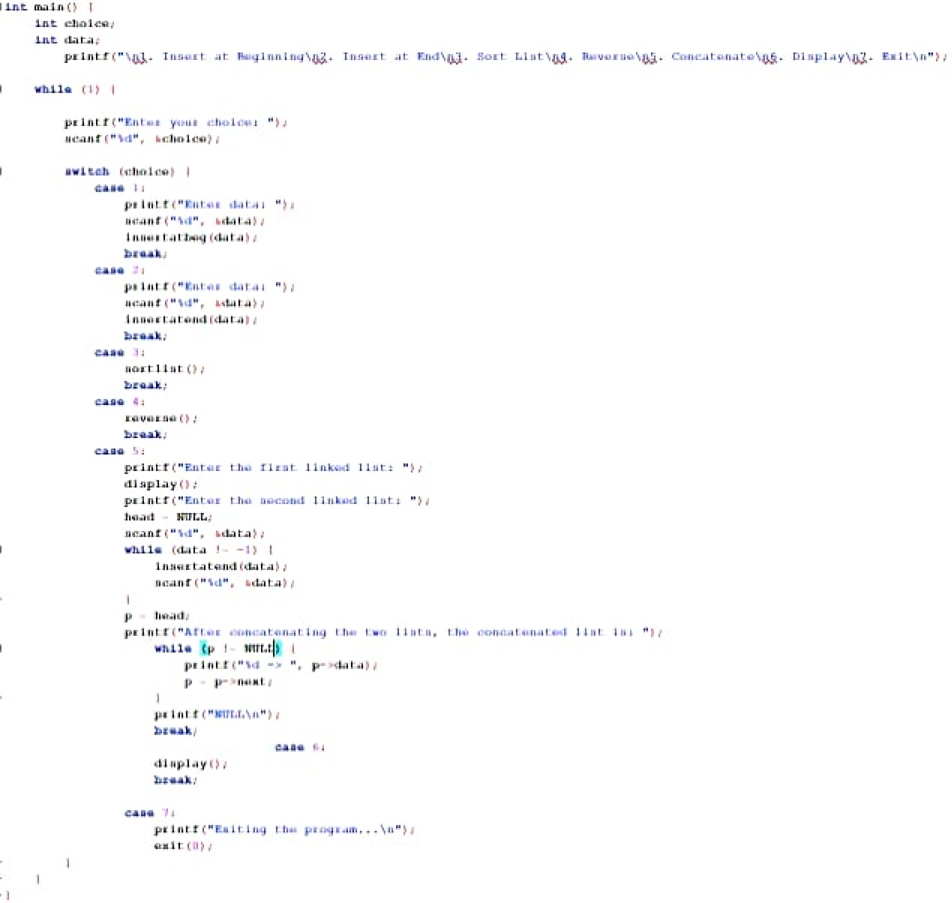
Output.

```
Enter choice 1 Enqueue 2 Dequeue 3. Display 4. Exit
1.
Enter the element  10
Enter choice  1. Enqueue 2. Dequeue 3. Display 4. Exit
1
Enter the element 20
Enter choice  1. Enqueue 2. Dequeue  3. Display 4. Exit
1 Enter the element 30
Enter choice   1. Enqueue   2. Dequeue  3. Display 4. Exit.
3.
10   20   30
Enter choice   1. Enqueue   2. Dequeue  3. Display
                                          4. Exit.
2
Enter choice   1. Enqueue  2. Dequeue  3. Display 4. Exit
1
Enter the element  40
Enter choice   1. Enqueue  2. Dequeue  3. Display 4. Exit.
2
Enter ~~the~~ choice  1. Enqueue  2. Dequeue  3. Display 4. Exit
3
    20   30   40
Enter choice   1. Enqueue  2. Dequeue  3. Display 4. Exit
4
```

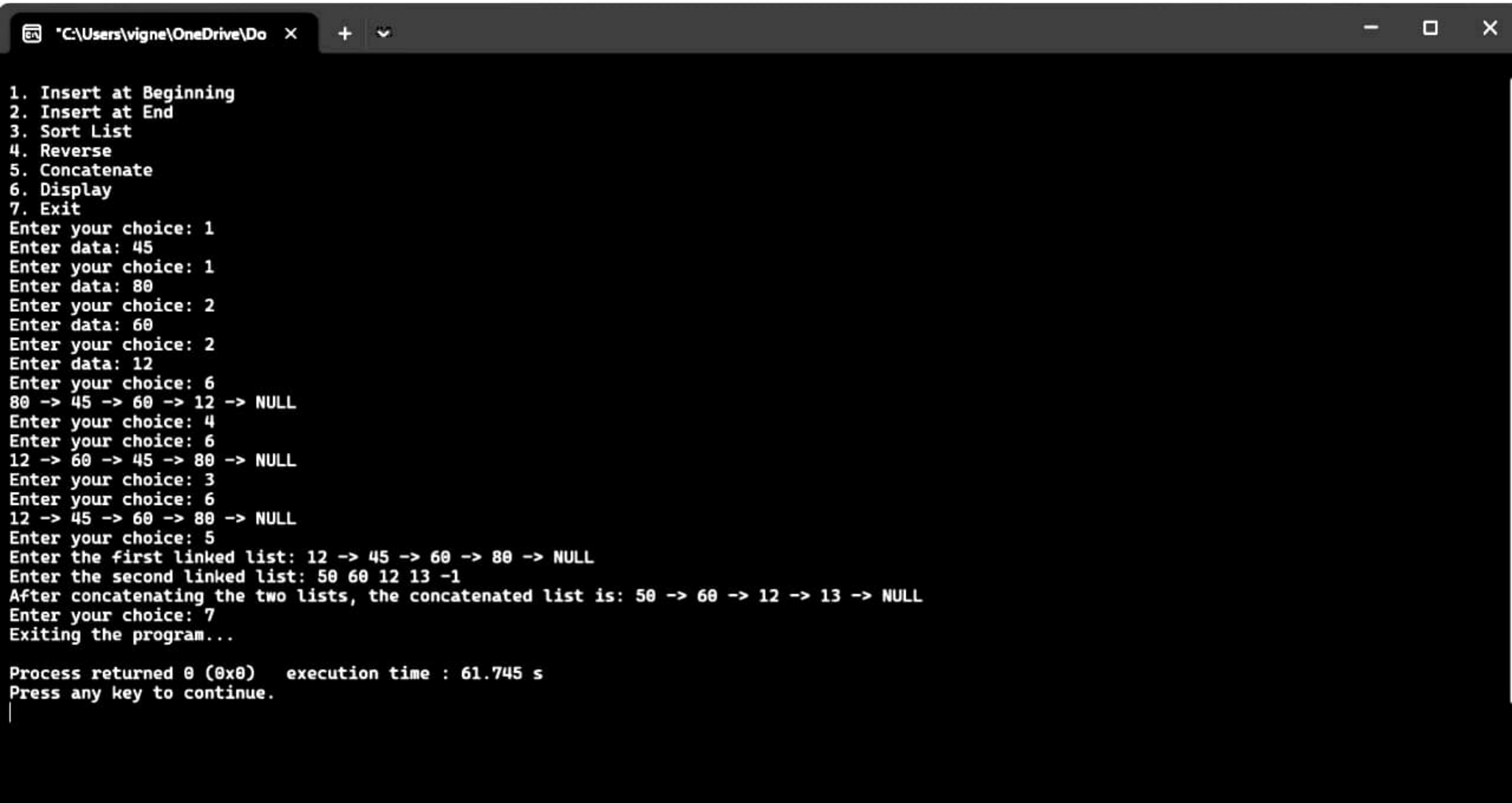


```
    current->next = newnode;
}

void concatenate(struct Node *p, struct Node *q) {
    if (head == NULL) {
        head = p;
    } else {
        struct Node *current = head;
        while (current->next != NULL) {
            current = current->next;
        }

        current->next = p;
    }

    while (q != NULL) {
        insertatend(q->data);
        q = q->next;
    }
}

void reverse() {
    prevnode = NULL;
    currentnode = head;
    newnode = NULL;

    while (currentnode != NULL) {
        newnode = currentnode->next;
        currentnode->next = prevnode;
        prevnode = currentnode;
        currentnode = newnode;
    }

    head = prevnode;
}
```

```
void sortlist() {
    i = head;
    while (i != NULL) {
        j = head;
        while (j->next != NULL) {
            if (j->data > j->next->data) {
                int temp = j->data;
                j->data = j->next->data;
                j->next->data = temp;
            }
            j = j->next;
        }
        i = i->next;
    }
}

void display() {
    struct Node *current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

int main() {
    int choice;
    int data;
        printf("\n1. Insert at Beginning\n2. Insert at End\n3. Sort List\n4. Reverse\n5. Concatenate\n6. Display\n7. Exit\n");

    while (1) {

        printf("Enter your choice: ");
        scanf("%d", &choice);
```



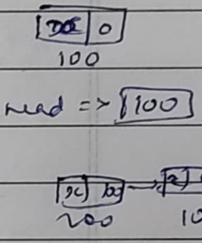
```
int main() {
    int choice;
    int data;
        printf("\n1. Insert at Beginning\n2. Insert at End\n3. Sort List\n4. Reverse\n5. Concatenate\n6. Display\n7. Exit\n");

    while (1) {

        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter data: ");
                scanf("%d", &data);
                insertatbeg(data);
                break;
            case 2:
                printf("Enter data: ");
                scanf("%d", &data);
                insertatend(data);
                break;
            case 3:
                sortlist();
                break;
            case 4:
                reverse();
                break;
            case 5:
                printf("Enter the first linked list: ");
                display();
                printf("Enter the second linked list: ");
                head = NULL;
                scanf("%d", &data);
                while (data != -1) {
                    insertatend(data);
                    scanf("%d", &data);
                }
                p = head;
                printf("After concatenating the two lists, the concatenated list is: ");
                    while (p != NULL) {
                        printf("%d -> ", p->data);
                        p = p->next;
                    }
                    printf("NULL\n");
                    break;
                                case 6:
                    display();
                    break;

                case 7:
                    printf("Exiting the program...\n");
                    exit(0);
        }
    }
}
```

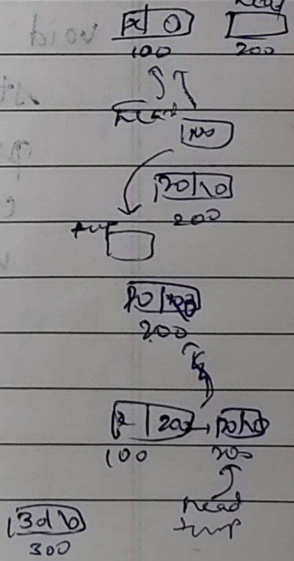


```
1. Insert at Beginning
2. Insert at End
3. Sort List
4. Reverse
5. Concatenate
6. Display
7. Exit
Enter your choice: 1
Enter data: 45
Enter your choice: 1
Enter data: 80
Enter your choice: 2
Enter data: 60
Enter your choice: 2
Enter data: 12
Enter your choice: 6
80 -> 45 -> 60 -> 12 -> NULL
Enter your choice: 4
Enter your choice: 6
12 -> 60 -> 45 -> 80 -> NULL
Enter your choice: 3
Enter your choice: 6
12 -> 45 -> 60 -> 80 -> NULL
Enter your choice: 5
Enter the first linked list: 12 -> 45 -> 60 -> 80 -> NULL
Enter the second linked list: 50 60 12 13 -1
After concatenating the two lists, the concatenated list is: 50 -> 60 -> 12 -> 13 -> NULL
Enter your choice: 7
Exiting the program...

Process returned 0 (0x0)   execution time : 61.745 s
Press any key to continue.
```

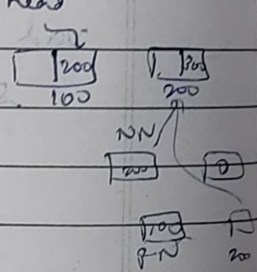
1. Sort, reverse, concatination using SLL

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

struct node {
      int data;
      struct node * next;
   };
 struct node * head = NULL;

void insertAtBeg (int ~~data~~ x) {
   struct node * newnode = (struct node *) malloc
                                  (sizeof (struct node));

     newnode -> data = x;
     newnode -> next = head;
       head = newnode;

  }
 void insertAtEnd (int x) {
     struct node * newnode = (struct node *)
                       malloc(sizeof(struct node));
       newnode -> data = x;
            ~~head~~
         newnode -> next = NULL;
       if (head == NULL)
      {
         head = newnode;
       }
       else {
          struct node * temp = head;
          while (temp -> next != NULL)
          {
           temp = temp -> next;
```





```
      }
    temp->next = newnode;
  }
}
~~void insert~~

void sortlist(){
   if (head ==NULL)
   {
      return;
   }
  struct node *i, *j;
  int temp;
  for (i= head; i->next != NULL ; i=i->next)
   {
   for(j=i->next ; j!=NULL; j=j->next
   {
        if (i->data > j->data){
        temp = i->data;
         i->data = j->data;
        j->data =temp;
      }
   }
  }
}
void reverse(){
   struct node * prev, *currentnode, *newnode;
   prevnode =0;
   currentnode = nextnode = head
   while(nextnode != 0)
   {
      nextnode = nextnode->next;
      currentnode->next = prevnode;
      prevnode= currentnode;
   } currentnode = nextnode;
  } head = prevnode;
```



```
~~void display()~~
void concatenate (struct node *p, struct node * q)
  {
    while (p!=NULL)
    {
        p = p->next;
    }
     p->next = q;
    return head;
   }


int main(){
   int choice, data;

   ~~while(1){~~
     printf("1Insert at Beginning \n");
     printf("2Insert at End");
     printf("3. Sort list);
      printf("4. Reverse list");
while(1){ printf("5. Concatenate");
      printf("Enter choice");
~~while(1){~~ scanf("%d", &choice);
      switch(choice){
        case 1:
          printf("Enter data at beginning");
          scanf("%d", &data);
           insertAtBeg(data);
            break
      case2;  printf("Enter data at end");
              scanf("%d", &data);
              insertAtEnd(data);
             break;
```

Date________ Page________

```
case 3 :   sortList( );
           ~~create~~
            display( );
            break;

case 4 :   reverse~~List~~( ) ;
             display( );
              break;


case 5:   struct node  *p , *q  ;
        int i=0;
        printf ("Insert first list);
      for (i=0 ; i<4; i++)
        scanf ("%d  &~~data~~ p→data);
        ~~scanf ("%d", & p→data);~~
          insertAtBeq (p→data)
        }

        printf ("Insert second list);
        for (i=0, i<4; i++)
          {
           scanf ("%d  , & q→data) ;
            insertAtBeq (q→data);
          }
      int x=concatenate ( &p , &q)
            ~~display~~ printf (" %d" ~~display~~, x);
   display(); break;
 Case 6:     ~~exit(0)~~; display ( );
 case 7:  exit (0);
}
 return 0;
}
```

29.01.24

Output:-

```
1. Insert at Beginning
2 Insert at End
3. Sort list
4. Reverse list
5. Concatenate
6. Display            7. exit.
  Enter choice
    1
  Enter data at beginning   20
  Enter choice
    1
  Enter data at beginning   30
  Enter choice
    2
  Enter data at End   50
  Enter choice
    3                                    // sorting
  Enter choice
    6                                    // display
   20    30    50
  Enter choice
    4                                    // reversing
  Enter choice
    6
   50    30    20
  Enter choice
  5
  Insert first list                      // concatenation
   10   30   5   7
   Insert second list
    20   40   50   96
   10   30   5  7   20   40   50   96
  Enter choice
    7
```

01.02.24