```
    temp = front;
    front = front->next;
    free(temp);
}
```

(6.)
```
#include<stdio.h>
#include<stdlib.h>
struct Node
{ int data;
  struct Node *next; };
struct Node *head1 = NULL;
struct Node *head2 = NULL;
struct Node * Createlist(int n)
void displayList(struct Node*head)
```

```
void sortList(struct Node *head)
{
    struct Node *i, *j;
    int tempData;
    if(head==NULL)
    {
        printf("List is empty\n");
        return;
    }
    for(i=head; i->next!=NULL; i=i->next)
    {
        for(j=i->next; j!=NULL; j=j->next)
        {
            if(i->data > j->data)
            {
                tempData = i->data;
                i->data = j->data;
                j->data = tempData;
            }
        }
    }
    printf("sorted linked list\n");
}
struct Node *reverseList(struct Node *head)
{
    struct Node *prev=NULL, *curr=head, *next=NULL;
    if(head==NULL)
    {
        printf("List is empty\n");
        return;
    }
    while(curr!=NULL)
    {
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
    printf("linked list reversed\n");
    return prev;
}
struct Node *concatList(struct Node *head1, struct Node *head2)
{
    struct Node *temp;
    if(head1==NULL) return head2;
    if(head2==NULL) return head1;
    temp = head1;
    while(temp->next!=NULL)
        temp = temp->next;
    temp->next = head2;
    printf("two lists got concatenated\n");
    return head1;
}
```

```
void main()
{
    int n1, n2, ch;
    printf("1. create linked list first, 2. create second list,
        3. display list1, 4. display list2, 5. sort list,
        6. reverse list, 7. concat list, 8. display concat list\n");
    do{
        printf("enter choice:\n");
        scanf("%d", &ch);
        switch(ch)
        {
            case 1: printf("enter number of nodes: ");
                scanf("%d", &n1);
                head1 = createList(n1);
                break;
            case 2: printf("enter number of nodes: ");
                scanf("%d", &n2);
                head2 = createList(n2);
                break;
            case 3: displayList(head1);
                break;
            case 4: displayList(head2);
                break;
            case 5: sortList(head1);
                break;
            case 6: head1 = reverseList(head1);
                break;
            case 7: head1 = concatList(head1, head2);
                head2 = NULL;
                break;
            case 8: displayList(head1);
                break;
            default: printf("invalid choice\n");
        }
    }while(ch!=0);
    return 0;
}
```

Output:

1. create first list, 2. creat second list, 3. display list1, 4. display list2
5. sort list, 6. reverse list, 7. concat list, 8. display concat list

enter choice: 1
enter number of nodes: 3
enter data: 30
enter data: 10

enter data: 20
Created linked list
enter choice: 2
enter number of nodes: 3
enter data: 40
enter data: 50
enter data: 60
Created linked list
enter choice: 3
30->10->20 -> enter choice: 4
40->50->60-> enter choice: 5
sorted linked list
enter choice: 3
10->20->30-> enter choice: 6
linked list reversed
enter choice: 3
30->20->10-> enter choice: 7
two lists got concatenated
enter choice: 8
30->20->10->40->50->60-> enter choice: 0
Invalid choice.

6b)
```
#include<stdio.h>
#include<stdlib.h>
struct Node *top=NULL;
struct Node *front=NULL;
struct Node *rear=NULL;
struct Node *createNode(int val)
void push(int val)
{ struct Node *newNode = createNode(val);
  newNode->next=top;
  top=newNode;
  printf("pushed val onto stack\n");
}
void pop()
{ struct Node *temp= top;
  if(top==NULL)
  { printf("stack is empty\n");
    return;
  }printf("removed element is %d\n",top->data);
  top=top->next;
  free(temp);
}
```

```
void enqueue(int val)
{ struct Node *newNode=createNode(val);
  if(rear==NULL)
  { front=rear=newNode;
  }else{
    rear->next=newNode;
    rear=newNode;
  }printf("enqueued element in queue\n");
}
void dequeue()
{ struct Node *temp;
  if(front==NULL)
  { printf("queue is empty\n");
    return;
  }
  temp=front;
  printf("removed element is %d\n",front->data);
  front=front->next;
  if(front==NULL)
    rear=NULL;
  free(temp);
}
void displayQueue()
void displayStack()
int main()
{ int choice, value;
  printf("1.stack Push, 2. stack Pop, 3. Display stack,
      4. Queue Enqueue, 5. Queue dequeue, 6.
      display Queue, 7.exit\n");
  while(1)
  { printf("enter choice: ");
    scanf("%d",&choice);
    switch(choice)
    { case 1: printf("enter value to push: ");
              scanf("%d",&value);
              push(value);
              break;
      case 2: pop();
              break;
      case 3: displayStack();
              break;
```

```
        case 4: printf("enter value to enqueue:");
                scanf("%d", &value);
                enqueue(value);
                break;
        case 5: dequeue();
                break;
        case 6: displayQueue();
                break;
        case 7: exit(0);
        default: printf("Invalid choice\n");
    }
}
return 0;
}
```

output:

1. Stack push, 2. stack pop, 3. display stack, 4. Queue enqueue
5. Queue dequeue, 6. display Queue, 7. Exit.

enter choice: 1
enter value to push: 10
pushed val onto the stack
enter choice: 1
enter value to push: 20
enter choice: 1
enter value to push: 30
enter choice: 2
enter value to push: 40
enter choice: 2
removed element is 40
enter choice: 3
30 -> 20 -> 10 -> enter choice: 4
enter value to enqueue: 10
enter choice: 4
enter value to enqueue: 20
enter choice: 4
enter value to enqueue: 30
enter choice: 5
removed element is 10
enter choice: 6
20 -> 30 -> enter choice: 7
Invalid choice.