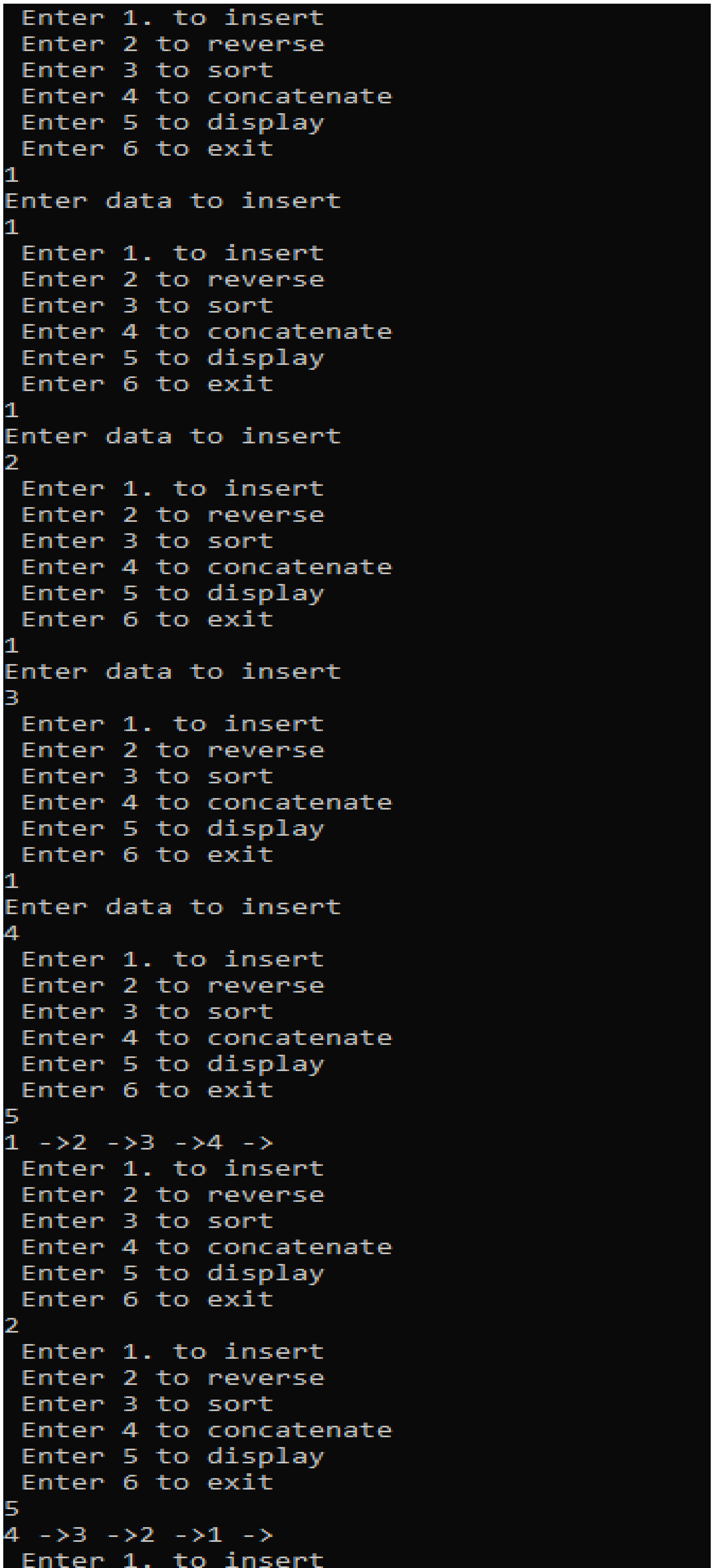
LAB 6

6a) WAP to Implement Single Link List with following operations: Sort the linked list, Reverse the linked list, Concatenation of two linked lists.

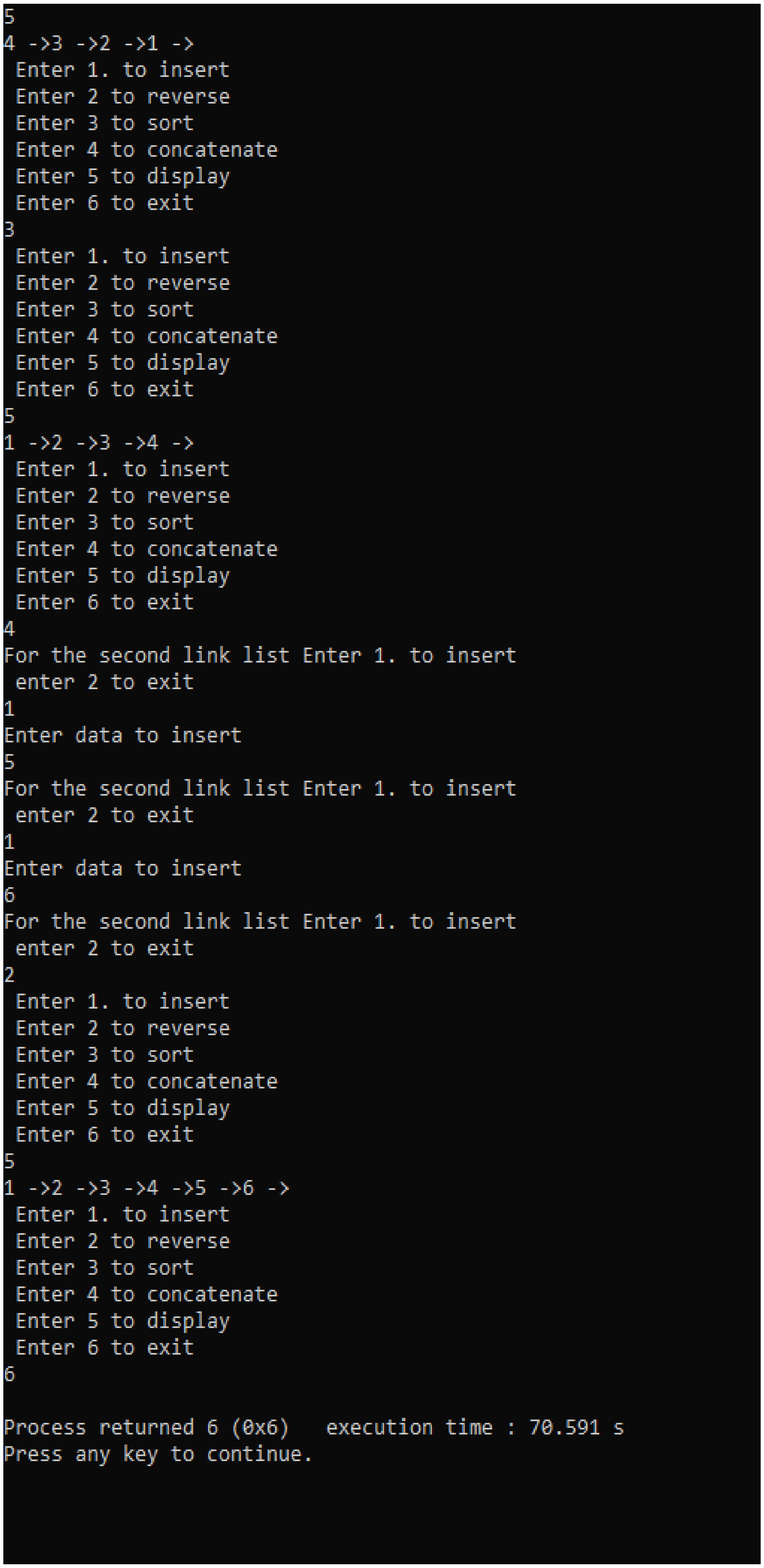
```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node* next;
};
struct node* head=NULL;
struct node* head2=NULL;
void insert(int item)
{
    struct node* newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=item;
    if(head==NULL)
    {
        head=newnode;
        newnode->next=NULL;

    }
    else
    {
        struct node* temp=head;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newnode;
        newnode->next=NULL;
    }
}
void insert2(int item)
{
    struct node* newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=item;
    if(head2==NULL)
    {
        head2=newnode;
        newnode->next=NULL;
    }
    else
    {
        struct node* temp=head2;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newnode;
        newnode->next=NULL;
    }
}
void reverse()
{
    struct node* curr=head;
    struct node* prev=NULL;
    struct node* next=NULL;
    while(curr != NULL)
    {
        next=curr->next;
        curr->next=prev;
        prev=curr;
        curr=next;
    }
    head=prev;
}
void sort()
{
    struct node* i;
    struct node* j;
    for(i=head;i!=NULL;i=i->next)
    {
        for(j=i->next;j!=NULL;j=j->next)
        {
            if(i->data>j->data)
            {
                int temp=i->data;
                i->data=j->data;
                j->data=temp;
            }
        }
    }
}
```

```
void concatenate()
{
   if(head==NULL)
   {
      head=head2;
   }
   struct node* temp=head;
   while(temp->next!=NULL)
   {
            temp=temp->next;
   }

   temp->next=head2;
}
void display()
{
   struct node* temp=head;
   while(temp!=NULL)
   {
      printf("%d ->",temp->data);
      temp=temp->next;
   }
   printf("\n");
}
void main()
{
   int choice=0;
   int item=0;
   int c=0;
   int i=0;
   while(choice!=6)
  {
   printf(" Enter 1. to insert\n Enter 2 to reverse \n Enter 3 to sort \n Enter 4 to concatenate \n
Enter 5 to display\n Enter 6 to exit\n");
   scanf("%d",&choice);
   switch(choice)
   {
   case 1:

      printf("Enter data to insert\n");
      scanf("%d",&item);
      insert(item);
      break;
   case 2:
      reverse();
      break;
   case 3:
      sort();
      break;
   case 4:

      while(c!=2)
      {
           printf("For the second link list Enter 1. to insert\n enter 2 to exit\n");
          scanf("%d",&c);
          if(c==1)
          {
            printf("Enter data to insert\n");
            scanf("%d",&i);
            insert2(i);
          }
      }
      concatenate();
      break;
   case 5:
      display();
      break;
   case 6:
      break;

   }
  }
}
```

OUTPUT:



```
 Enter 1. to insert
 Enter 2 to reverse
 Enter 3 to sort
 Enter 4 to concatenate
 Enter 5 to display
 Enter 6 to exit
1
Enter data to insert
1
 Enter 1. to insert
 Enter 2 to reverse
 Enter 3 to sort
 Enter 4 to concatenate
 Enter 5 to display
 Enter 6 to exit
1
Enter data to insert
2
 Enter 1. to insert
 Enter 2 to reverse
 Enter 3 to sort
 Enter 4 to concatenate
 Enter 5 to display
 Enter 6 to exit
1
Enter data to insert
3
 Enter 1. to insert
 Enter 2 to reverse
 Enter 3 to sort
 Enter 4 to concatenate
 Enter 5 to display
 Enter 6 to exit
1
Enter data to insert
4
 Enter 1. to insert
 Enter 2 to reverse
 Enter 3 to sort
 Enter 4 to concatenate
 Enter 5 to display
 Enter 6 to exit
5
1 ->2 ->3 ->4 ->
 Enter 1. to insert
 Enter 2 to reverse
 Enter 3 to sort
 Enter 4 to concatenate
 Enter 5 to display
 Enter 6 to exit
2
 Enter 1. to insert
 Enter 2 to reverse
 Enter 3 to sort
 Enter 4 to concatenate
 Enter 5 to display
 Enter 6 to exit
5
4 ->3 ->2 ->1 ->
 Enter 1. to insert
```

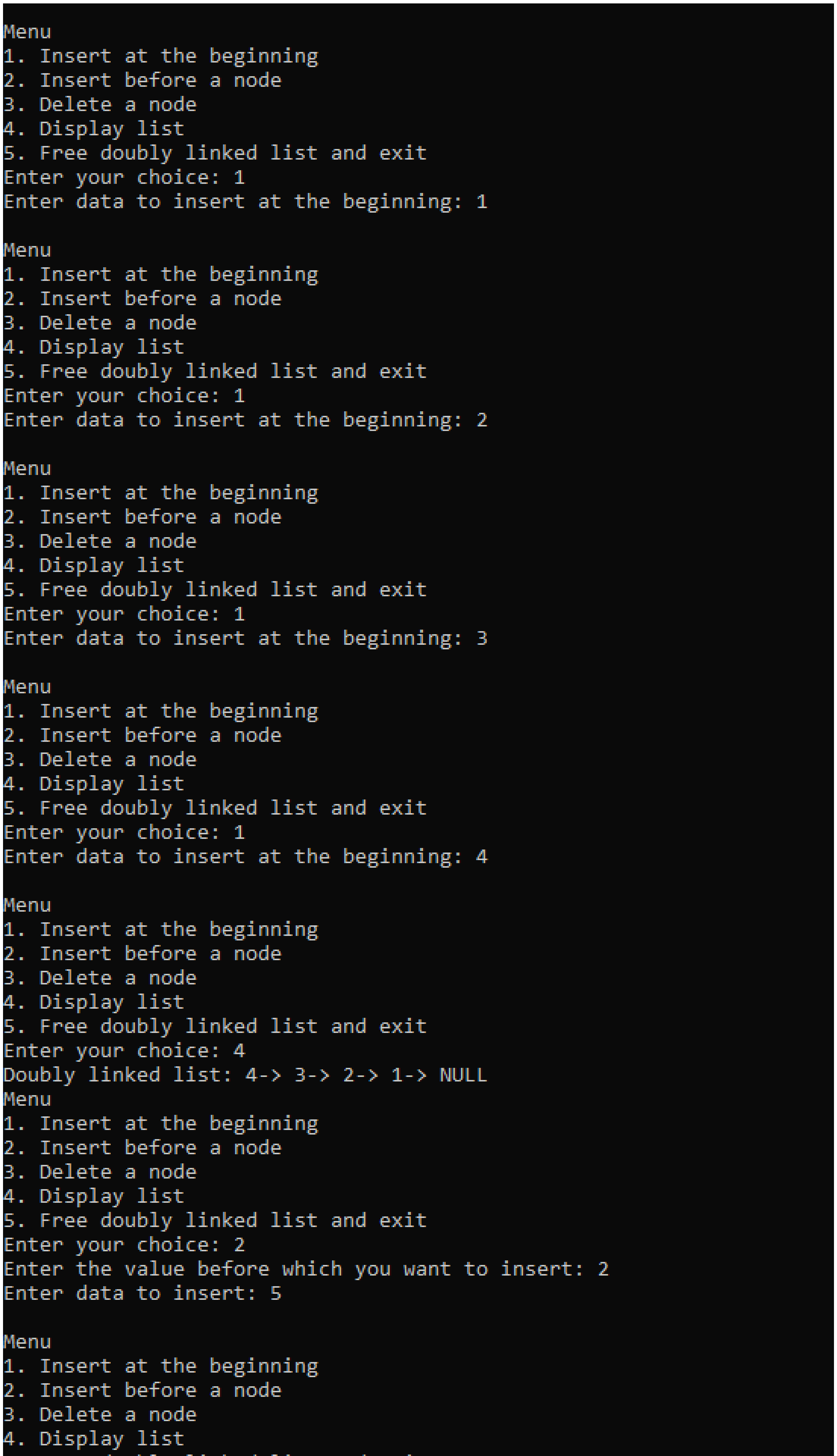


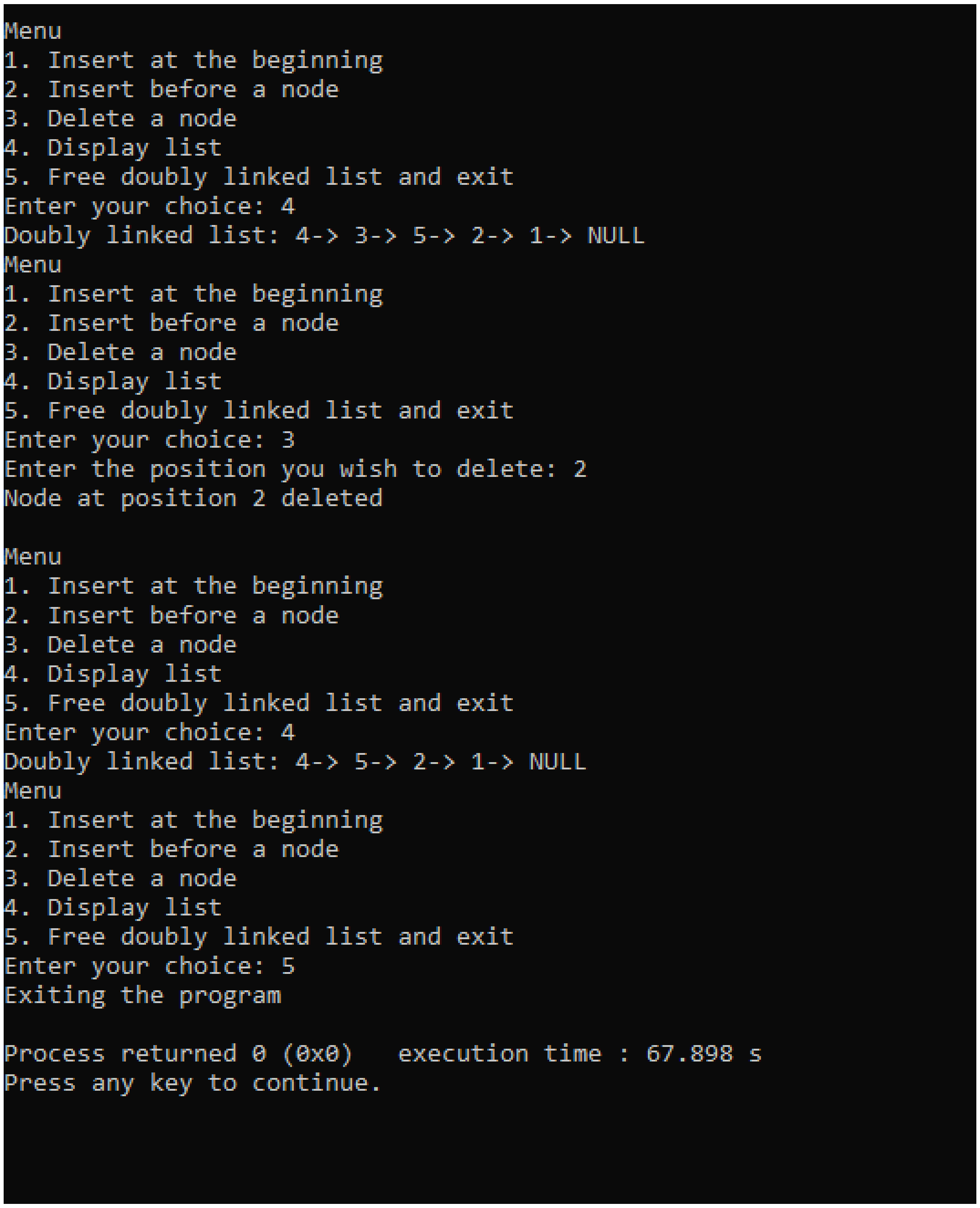
```
5
4 ->3 ->2 ->1 ->
 Enter 1. to insert
 Enter 2 to reverse
 Enter 3 to sort
 Enter 4 to concatenate
 Enter 5 to display
 Enter 6 to exit
3
 Enter 1. to insert
 Enter 2 to reverse
 Enter 3 to sort
 Enter 4 to concatenate
 Enter 5 to display
 Enter 6 to exit
5
1 ->2 ->3 ->4 ->
 Enter 1. to insert
 Enter 2 to reverse
 Enter 3 to sort
 Enter 4 to concatenate
 Enter 5 to display
 Enter 6 to exit
4
For the second link list Enter 1. to insert
 enter 2 to exit
1
Enter data to insert
5
For the second link list Enter 1. to insert
 enter 2 to exit
1
Enter data to insert
6
For the second link list Enter 1. to insert
 enter 2 to exit
2
 Enter 1. to insert
 Enter 2 to reverse
 Enter 3 to sort
 Enter 4 to concatenate
 Enter 5 to display
 Enter 6 to exit
5
1 ->2 ->3 ->4 ->5 ->6 ->
 Enter 1. to insert
 Enter 2 to reverse
 Enter 3 to sort
 Enter 4 to concatenate
 Enter 5 to display
 Enter 6 to exit
6

Process returned 6 (0x6)   execution time : 70.591 s
Press any key to continue.
```

8)WAP to Implement doubly link list with primitive operations
I.Create a doubly linked list.
II. Insert a new node to the left of the node.
III. Delete the node based on a specific value
IV. Display the contents of the list

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

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

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        return NULL;
    }
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

void insertAtBeginning(struct Node** head, int data) {
    struct Node* newNode = createNode(data);

    if (*head == NULL) {
        *head = newNode;
    } else {
        newNode->next = *head;
        (*head)->prev = newNode;
        *head = newNode;
    }
}
void insertBeforeNode(struct Node** head, int key, int data) {
    if (*head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node* newNode = createNode(data);
    struct Node* current = *head;

    while (current) {
        if (current->data == key) {
            if (current->prev) {
                current->prev->next = newNode;
                newNode->prev = current->prev;
            } else {
                *head = newNode;
            }

            newNode->next = current;
            current->prev = newNode;
            return;
        }
        current = current->next;
    }

    printf("Key not found in the list\n");
}


void deleteNode(struct Node** head, int pos) {
    if (*head == NULL) {
        printf("List is empty\n");
        return;
    }

    struct Node* current = *head;
    int count = 1;

    while (current && count < pos) {
        current = current->next;
        count++;
    }

    if (current == NULL) {
        printf("Position %d is beyond the length of the list\n", pos);
        return;
    }

    if (current->prev) {
        current->prev->next = current->next;
    } else {
        *head = current->next;
    }
    if (current->next) {
        current->next->prev = current->prev;
    }

    free(current);
    printf("Node at position %d deleted\n", pos);
}

void displayList(struct Node* head) {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }

    struct Node* current = head;

    while (current) {
        printf("%d-> ", current->data);
        current = current->next;
    }
    printf("NULL");
}

void freeList(struct Node* head) {
    struct Node* current = head;
    struct Node* nextNode;

    while (current) {
        nextNode = current->next;
        free(current);
        current = nextNode;
    }
}

int main() {
    struct Node* head = NULL;
    int ch, newData, pos, key;

    while (1) {
        printf("\nMenu\n");
        printf("1. Insert at the beginning\n");
        printf("2. Insert before a node\n");
        printf("3. Delete a node\n");
        printf("4. Display list\n");
        printf("5. Free doubly linked list and exit\n");
        printf("Enter your choice: ");
        scanf("%d", &ch);

        switch (ch) {
          case 1:
            printf("Enter data to insert at the beginning: ");
            scanf("%d", &newData);
            insertAtBeginning(&head, newData);
            break;

          case 2:
            printf("Enter the value before which you want to insert: ");
            scanf("%d", &key);
            printf("Enter data to insert: ");
            scanf("%d", &newData);
            insertBeforeNode(&head, key, newData);
            break;

          case 3:
            printf("Enter the position you wish to delete: ");
            scanf("%d", &key);
            deleteNode(&head, key);
            break;

          case 4:
            printf("Doubly linked list: ");
            displayList(head);
            break;

          case 5:
            freeList(head);
            printf("Exiting the program\n");
            return 0;

          default:
            printf("Invalid choice\n");
        }
    }

    return 0;
}
```

OUTPUT:


```
Menu
1. Insert at the beginning
2. Insert before a node
3. Delete a node
4. Display list
5. Free doubly linked list and exit
Enter your choice: 1
Enter data to insert at the beginning: 1

Menu
1. Insert at the beginning
2. Insert before a node
3. Delete a node
4. Display list
5. Free doubly linked list and exit
Enter your choice: 1
Enter data to insert at the beginning: 2

Menu
1. Insert at the beginning
2. Insert before a node
3. Delete a node
4. Display list
5. Free doubly linked list and exit
Enter your choice: 1
Enter data to insert at the beginning: 3

Menu
1. Insert at the beginning
2. Insert before a node
3. Delete a node
4. Display list
5. Free doubly linked list and exit
Enter your choice: 1
Enter data to insert at the beginning: 4

Menu
1. Insert at the beginning
2. Insert before a node
3. Delete a node
4. Display list
5. Free doubly linked list and exit
Enter your choice: 4
Doubly linked list: 4-> 3-> 2-> 1-> NULL
Menu
1. Insert at the beginning
2. Insert before a node
3. Delete a node
4. Display list
5. Free doubly linked list and exit
Enter your choice: 2
Enter the value before which you want to insert: 2
Enter data to insert: 5

Menu
1. Insert at the beginning
2. Insert before a node
3. Delete a node
4. Display list
```



```
Menu
1. Insert at the beginning
2. Insert before a node
3. Delete a node
4. Display list
5. Free doubly linked list and exit
Enter your choice: 4
Doubly linked list: 4-> 3-> 5-> 2-> 1-> NULL
Menu
1. Insert at the beginning
2. Insert before a node
3. Delete a node
4. Display list
5. Free doubly linked list and exit
Enter your choice: 3
Enter the position you wish to delete: 2
Node at position 2 deleted

Menu
1. Insert at the beginning
2. Insert before a node
3. Delete a node
4. Display list
5. Free doubly linked list and exit
Enter your choice: 4
Doubly linked list: 4-> 5-> 2-> 1-> NULL
Menu
1. Insert at the beginning
2. Insert before a node
3. Delete a node
4. Display list
5. Free doubly linked list and exit
Enter your choice: 5
Exiting the program

Process returned 0 (0x0)   execution time : 67.898 s
Press any key to continue.
```