LAB-07

Date: 05-02-24
Page No:

1. WAP to implement doubly link list with primite operations: a) Create a doubly linked list
b) Insert a new node to the left of the node
c) Delete the node based on a specific value.

```
#include <stdio.h>

struct node{
    int data;
    struct node *next;
    struct node *prev;
};
struct node *head=0, *newnode, *temp;

void create()
{ int i,n;
  printf("Enter the no. of elements:\n");
  scanf("%d", &n);

  for(i=0; i<n; i++)
  { newnode=(struct node*) malloc(sizeof(struct node));
    printf("Enter the %d element:\n", i+1);
    scanf("%d", &newnode->data);
    newnode->prev=0;
    newnode->next=0;

    if(head==0)
    { temp=head=newnode; }
    else
    { temp->next=newnode;
      newnode->prev=temp;
      temp=newnode;
    }
}
```

```
void display()
{ temp=head;
  while(temp!=0)
  { printf("%d\n", temp->data);
    temp= temp->next;
  }
}

void insert_left()
{ int node, i=1;
  printf("enter the node\n");
  scanf("%d", &node);
  temp=head;
  if(node<1)
  { printf("invalid position\n"); }
  else if(node==1)
  {
   newnode=(struct node *)malloc(sizeof(struct node));
   printf("enter data\n");
   scanf("%d", &newnode->data);
   newnode->prev=0;
   head->prev=newnode;
   newnode->next=head;
   head=newnode;
  }
 else
 {newnode=(struct node *)malloc(sizeof(struct node));
  printf("enter data\n");
  scanf("%d", &newnode->data);
  while(i<node-1)
  { temp=temp->next;
    i++;
  }
```

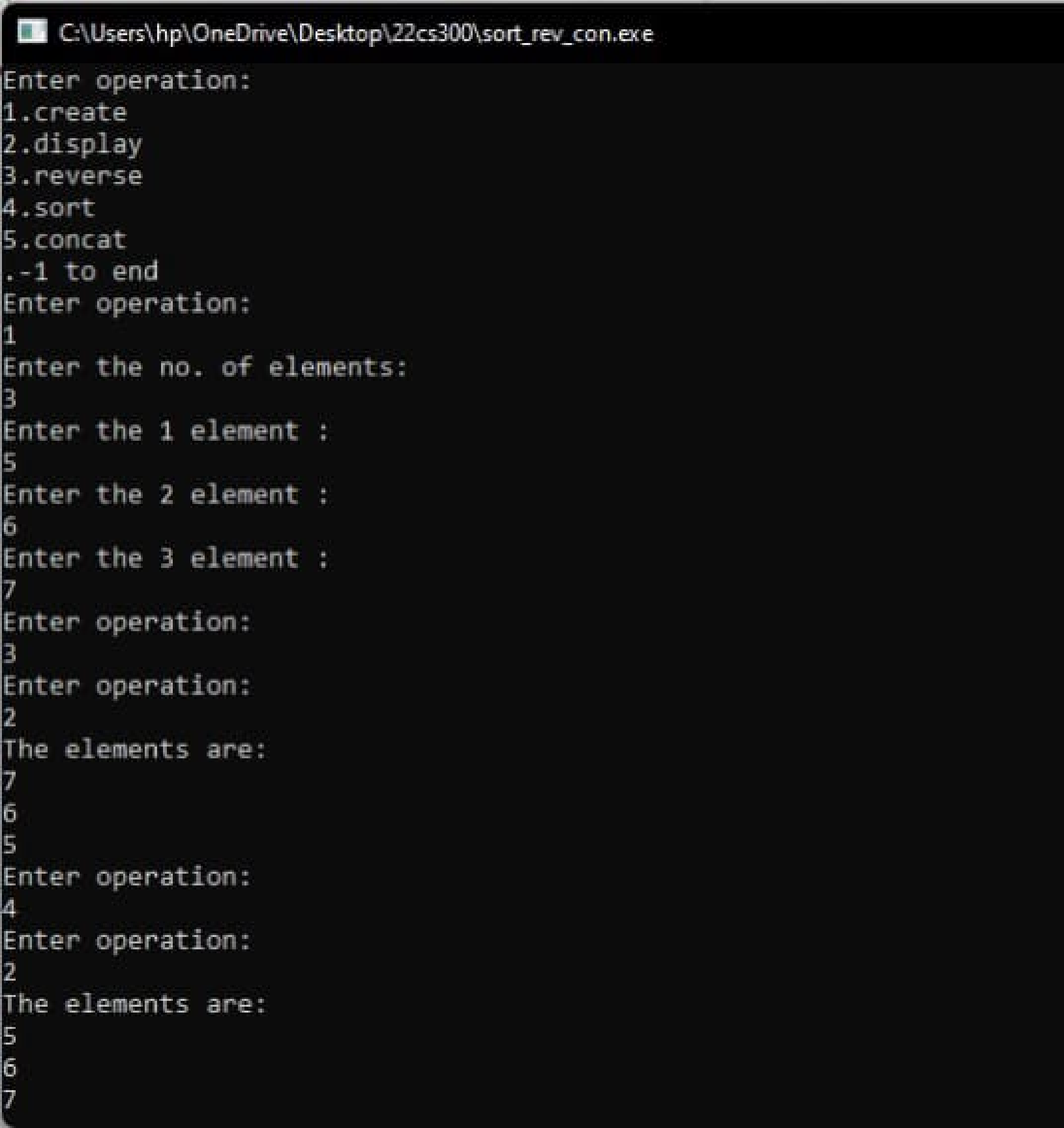
```
        newnode->prev = temp;
        newnode->next = temp->next;
        temp->next = newnode;
        newnode->next->prev = newnode;
      }
}

void delet_pos()
{ int pos, i = 1;
  temp = head;
  printf("enter position\n");
  scanf("%d", &pos);
  while(i<pos)
  {
    temp = temp->next;
    i++;
  }
  temp->prev->next = temp->next;
  temp->next->prev = temp->prev;
  free(temp);
}

void main()
{ int choice;
  printf("enter operation\n1.create\n2.display
        \n3.insert to the left\n4.delete from a
        position\n5.-1 to stop\n");
  while(1)
  { printf("Enter operation\n");
    scanf("%d", &choice);
```

```
if(choice==-1)
{printf("operation completed\n");
 break;
}
else
{switch(choice)
 { case 1: create();
           break;
  case 2: display();
           break;
  case 3: insert_left();
           break;
  case 4: delete_pos();
           break;
  default: printf("invalid input\n");
 }
}
}
}
```

//output:

Enter operation
1. create
2. display
3. insert to left
4. delete from a position
5. -1 to stop
enter operation: 1
enter the no. of elements 3
enter the 1 element: 5
enter the 2 element: 6
enter the 3 element: 7

enter Operation 2
5
6
7
enter operation 3
enter the node 1
enter data 8
enter operation 2
8
5
6
7
enter operation 3
enter the node 3
Enter data 9
enter operation 2
8
5
9
6
7
enter operation
4
enter position 4
enter operation 2
8
5
9
7
enter operation -1
operation completed


C:\Users\hp\OneDrive\Desktop\22cs300\sort_rev_con.exe
Enter operation:
1.create
2.display
3.reverse
4.sort
5.concat
.-1 to end
Enter operation:
1
Enter the no. of elements:
3
Enter the 1 element :
5
Enter the 2 element :
6
Enter the 3 element :
7
Enter operation:
3
Enter operation:
2
The elements are:
7
6
5
Enter operation:
4
Enter operation:
2
The elements are:
5
6
7