LAB-05

Date : 22-01-24
Page No :

1. WAP to implement singly linked list with following operations: a) create a linked list b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list.

Program

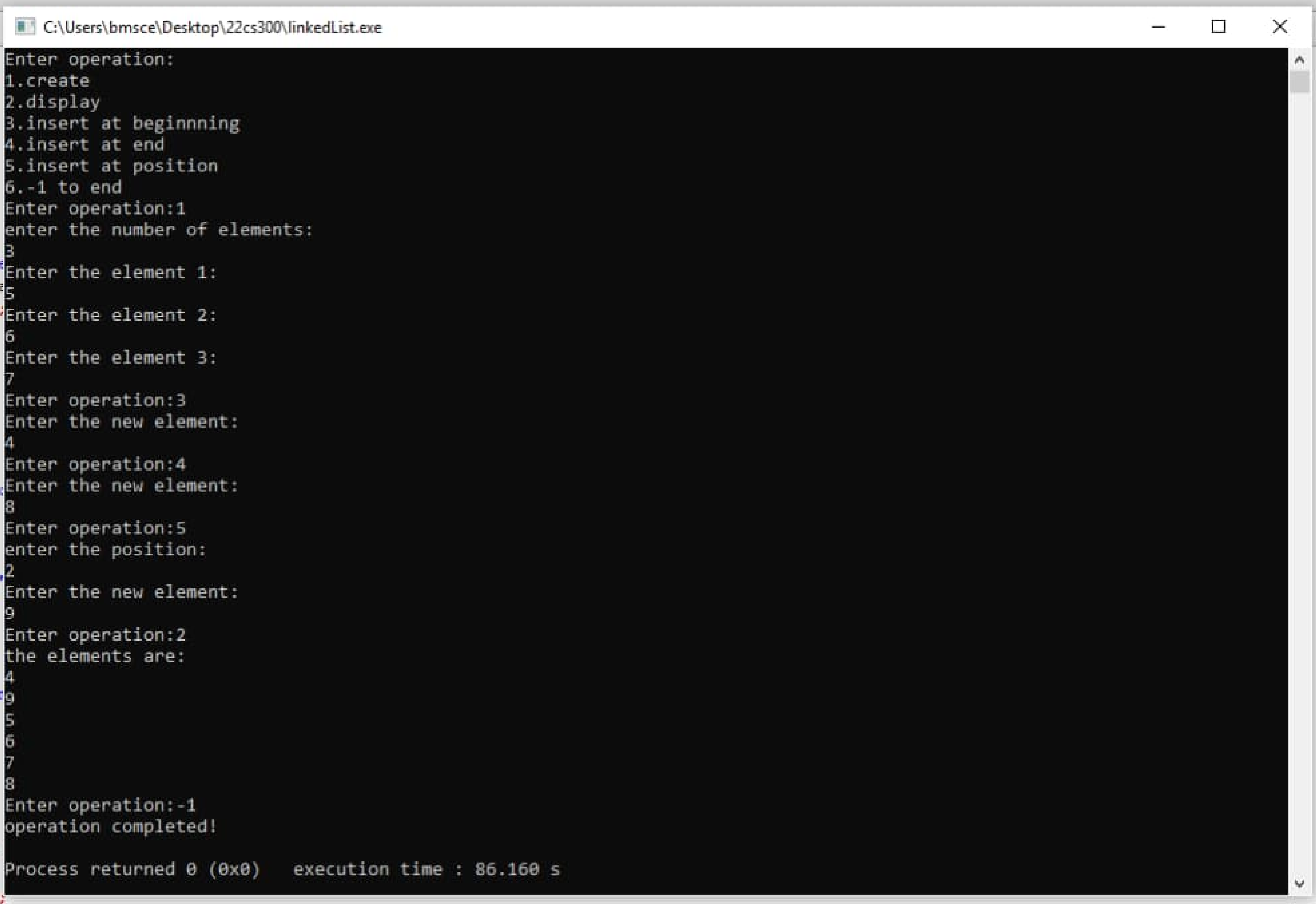
```
#include<stdio.h>

struct node{
        int data;
        struct node * next;
};
struct node *head=NULL, *newnode, *temp;
~~int~~
void create()
{
    int i, n;
    printf("Enter the no. of elements:\n");
    scanf("%d", &n);

22/1/24
    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->next=NULL;

        if(head==NULL)
        { temp=head=newnode;
        }
        else
        { temp→next = newnode ;
          temp = newnode ;
        }
    }
}

void display()
{
   temp = head ;
   printf("The elements are:\n");
   while(temp != NULL)
    {
        printf("%d\n", temp→data);
        temp = temp→next;
    }
}

void insert_beg()
{
  newnode = (struct node *) malloc (sizeof(struct no
  printf("enter the new element:\n");
  scanf("%d", &newnode→data);
  newnode→next = head;
  head = newnode;
}

void insert_end()
{
  newnode = (struct node *)malloc(sizeof(struct no
  printf("Enter the new element\n");
  scanf("%d", &newnode→data);
```

```
        newnode->next=NULL;
        temp=head;
        while (temp->next!=NULL)
         {
            temp=temp->next;
         }
         temp->next=newnode;
}

void insert_pos()
{
    int pos, i=0;
    newnode=(struct node *)malloc(sizeof(struct node));
    printf("Enter the position:\n");
    scanf("%d", &pos);

    if(pos<0)
     {
        printf("invalid position\n");
     }
    else
     {
        temp=head;
        while(i<pos-1)
         {
            temp=temp->next;
            i++;
         }
        printf("Enter the new element\n");
        scanf("%d", &newnod->data);
        newnode->next=temp->next;
        temp->next=newnode;
     }
}
```

```
void main()
{
    int choice;
    while(1)
    {
      printf("Enter operation:\n1.create\n2.display
              \n3.insert at beginning\n4.insert at
             end\n5.insert at a position\n6.-1 to
              end\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_beg()
                     break;
           case 4 :insert_end()
                     break;
           case 5 : insert_pos();
                     break;
          default: printf("invalid input\n");
         }
      }
}
```

Date :
Page No :

//Output
```
Enter operation:
1. create 2. display 3. insert at beginning
4. Insert at end 5 insert at position. 6 -1 end
Enter operation: 1
enter the number of elements: 3
Enter the element 1: 5
Enter the element 2: 6
Enter the element 3: 7
Enter operation: 2
the elements are: 5
                  6
                  7

Enter operation: 3
Enter the new element: 4
Enter operation: 4
Enter the new element: 8
Enter operation: 5
enter the position: 2
enter the new element: 9

Enter operation: 2
the elements are: 4
                  9
                  5
                  6
                  7
                  8
Enter operation: -1
operation completed!
```



```
Enter operation:
1.create
2.display
3.insert at beginnning
4.insert at end
5.insert at position
6.-1 to end
Enter operation:1
enter the number of elements:
3
Enter the element 1:
5
Enter the element 2:
6
Enter the element 3:
7
Enter operation:3
Enter the new element:
4
Enter operation:4
Enter the new element:
8
Enter operation:5
enter the position:
2
Enter the new element:
9
Enter operation:2
the elements are:
4
9
5
6
7
8
Enter operation:-1
operation completed!

Process returned 0 (0x0)   execution time : 86.160 s
```

2. WAP to implement singly linked list with following operations: a) create a linked list b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list.

program

```
#include <stdio.h>

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

struct node *head=NULL, *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->next=NULL;

       if (head==NULL)
        { temp=head=newnode;
        }
        else {
                temp->next = newnode;
                temp = newnode;
              }
        }
}
void display ()
{
      temp = head;
      printf("The elements are:\n");
      while (temp != NULL)
        {
              printf("%d\n", temp->data);
              temp = temp->next;
        }
  }



void delete_beg()
 {
     temp = head;
     if (head == NULL)
       {
            printf(" List is empty");
       }
     else
      {
            head = temp->next;
            free(temp);
      }
 }
```

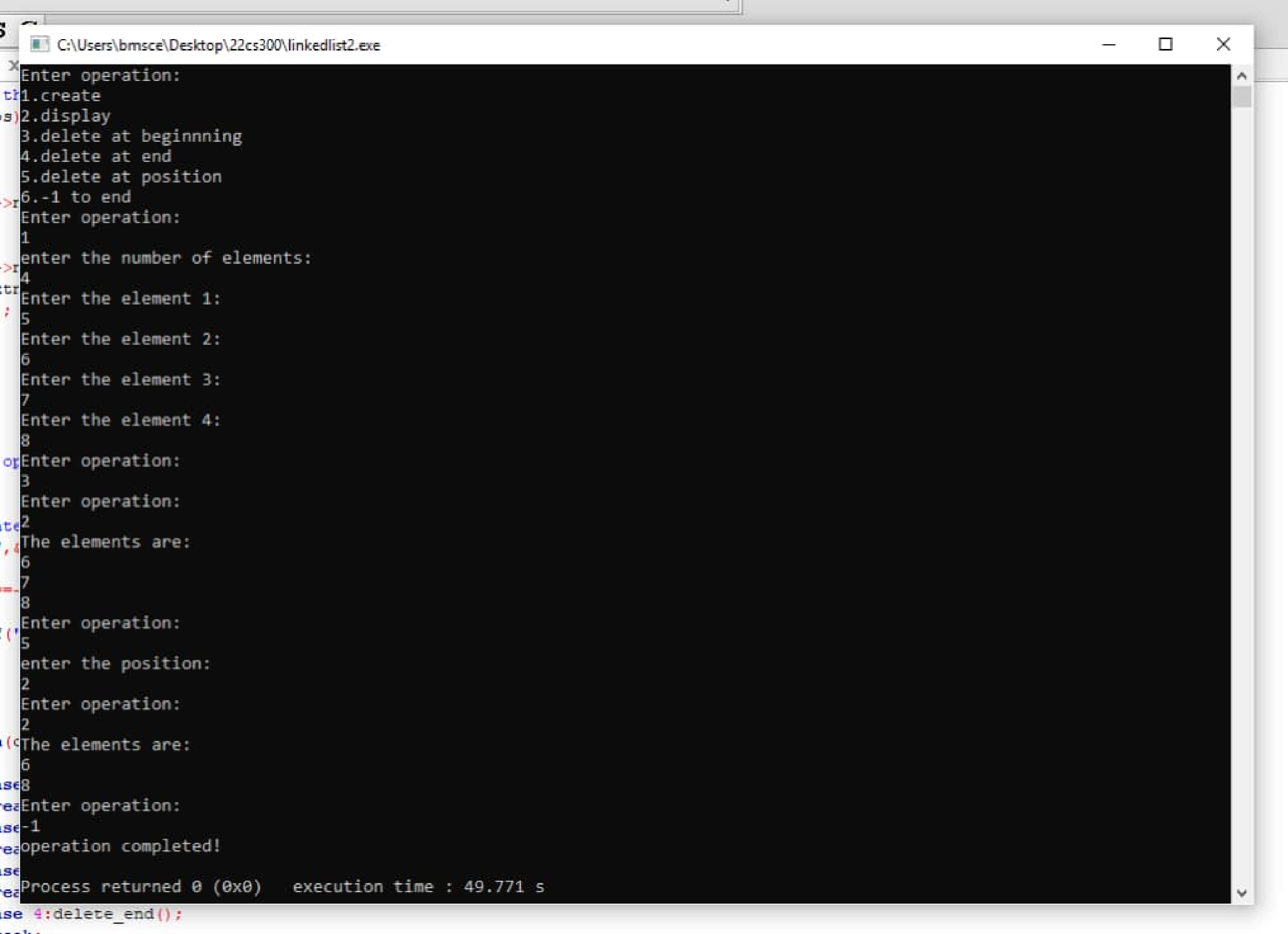
```
void delete_end(){
        struct node *prenode;
        temp=head;
        while(temp→next!=NULL)
          {
              prenode=temp;
              temp=temp→next;
          }
        if(temp=head)
          { head=NULL; }
        else
           { prenode→next=NULL;}
        free(temp);
   }
void delete_pos()
{  struct node *nextnode;
   int pos,i=1;
   temp=head;
   printf("enter position\n");
   scanf("%d",&pos);
   while(i<pos)
     {
         temp=temp→next;
         i++;
     }
     nextnode=temp→next;
     temp→next=nextnode→next;
     free(nextnode);
}
void main()
{
     int choice;
while (1)
  { printf("Enter operation:\n1.create\n2.display
         \n3.delete at beginning\n4.delete at
         end\n5.delete at a position\n6.-1 to
         end\n");
   scanf("%d", &choice);
   if (choice ==-1)
   { printf("operation completed!\n");
     break;
   }
   else { switch(choice)
          {
            case1: create();
                   break;
            case2: display();
                   break;
            case 3: delete_beg()
                   break;
            case 4: delete_end()
                   break;
            case 5: delete_pos()
                   break;
           default: printf("invalid input\n");
          }
        }
     }
 }
```

//output

```
Enter operation:
1.create
2.display
3.delete at beginning
```

```
4. delete at end
5- delete at position
6. -1 to end
Enter operation : 1
enter the number of elements : 4
Enter the element 1 : 5
Enter the element 2 : 6
Enter the element 3 : 7
Enter the element 4 : 8

Enter operation : 2
The elements are : 5
                 : 6
                 : 7
                 : 8

Enter operation: 3
Enter operation : 2
The elements are : 6
                   7
                   8

Enter operation: 4
Enter operation : 2
The elements are : 6
                   7

Enter operation : 5
Enter the position : 2
Enter operation : 2
The elements are : 6
~~En~~'
Enter operation : -1
operation completed !
```

[illegible] 22/11/24


C:\Users\bmsce\Desktop\22cs300\linkedlist2.exe

```
Enter operation:
1.create
2.display
3.delete at beginnning
4.delete at end
5.delete at position
6.-1 to end
Enter operation:
1
enter the number of elements:
4
Enter the element 1:
5
Enter the element 2:
6
Enter the element 3:
7
Enter the element 4:
8
Enter operation:
3
Enter operation:
2
The elements are:
6
7
8
Enter operation:
5
enter the position:
2
Enter operation:
2
The elements are:
6
8
Enter operation:
-1
operation completed!

Process returned 0 (0x0)   execution time : 49.771 s
```

```
se 4:delete_end();
```