# /*circular linked list full operation*/

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
	int data;
	struct node*link;
};
struct node*header;
struct node*create_cll(struct node*);
struct node*display(struct node*);
struct node*insert_beg(struct node*);
struct node*insert_end(struct node*);
struct node*insert_any(struct node*);
struct node*delete_beg(struct node*);
struct node*delete_end(struct node*);
struct node*delete_any(struct node*);
struct node*sort_list(struct node*);
int main()
{
	int choice=0;
	while(choice!=10)
	{
		printf("**main menu**\n");
		printf("1.create list\n2.display the list\n3.insert at the begining\n4.insert at the end\n5.insert at any position\n6.delete from the begining\n7.delete from the end\n8.delete from any position\n9.sort the list\n10.exit\n");
		printf("enter your choice\n");
		scanf("%d",&choice);
		switch(choice)
		{
```

```
                        case 1:header=create_cll(header);
                        break;
                        case 2:header=display(header);
                        break;
                        case 3:header=insert_beg(header);
                        break;
                        case 4:header=insert_end(header);
                        break;
                        case 5:header=insert_any(header);
                        break;
                        case 6:header=delete_beg(header);
                        break;
                        case 7:delete_end(header);
                        break;
                        case 8:delete_any(header);
                        break;
                        case 9:header=sort_list(header);
                        break;
                        case 10:exit(0);
                        default:
                                printf("invalid choice\n");
                }
        }
}
struct node*create_cll(struct node*header)
{
        struct node*new_node,*ptr;
        int item;
        printf("enter -1 to end\n");
        printf("enter the data: \n");
        scanf("%d",&item);
```

```
	while(item!=-1)
	{
		new_node=(struct node*)malloc(sizeof(struct node*));
		new_node->data=item;
		if(header==NULL)	//list is empty
		{
			new_node->link=new_node;
			header=new_node;
		}
		else
		{
			ptr=header;
			while(ptr->link!=header)
			{
				ptr=ptr->link;
		   }
				ptr->link=new_node;
				new_node->link=header;
		}
		printf("enter the data: \n");
		scanf("%d",&item);
	}
	printf("link list is created\n");
	return header;
}
struct node*display(struct node*header)
{
	printf("the linked list is below\n");
	struct node*ptr;
	ptr=header;
	while(ptr->link!=header)	//list is not empty
```

```
        {
                printf("%d\n",ptr->data);
                ptr=ptr->link;
   }
   printf("%d\n",ptr->data);       //as the link of lat node is header,so the loop wil stop but we have to print the value of the node
   return header;
}
struct node*insert_beg(struct node*header)
{
        struct node*new_node,*ptr;
        int item;
        if(header==NULL)
        {
                printf("overflow:insertion not possible\n");       //memory bank returns NULL
        }
        else
        {
                printf("enter the data to be inserted: \n");
           scanf("%d",&item);
                new_node=(struct node*)malloc(sizeof(struct node*));
                new_node->data=item;
                ptr=header;
                while(ptr->link!=header)
                {
                        ptr=ptr->link;
                }
                ptr->link=new_node;
                new_node->link=header;
                header=new_node;
                printf("node inserted at the end\n");
```

```
                return header;
        }
}
        /*struct node*new_node;
        int item;
        if(header==NULL)
        {
                printf("overflow:insertion not possible\n");      //memory bank returns NULL
        }
        else
        {
                printf("enter the data to be inserted: \n");
           scanf("%d",&item);
                new_node=(struct node*)malloc(sizeof(struct node*));
                new_node->data=item;
                new_node->link=header;
                header=new_node;
                printf("node inserted at the begining\n");
                return header;
        }
        */
struct node*insert_end(struct node*header)
{
        struct node*new_node,*ptr;
        int item;
        if(header==NULL)
        {
                printf("overflow:insertion not possible\n");      //memory bank returns NULL
        }
        else
        {
```

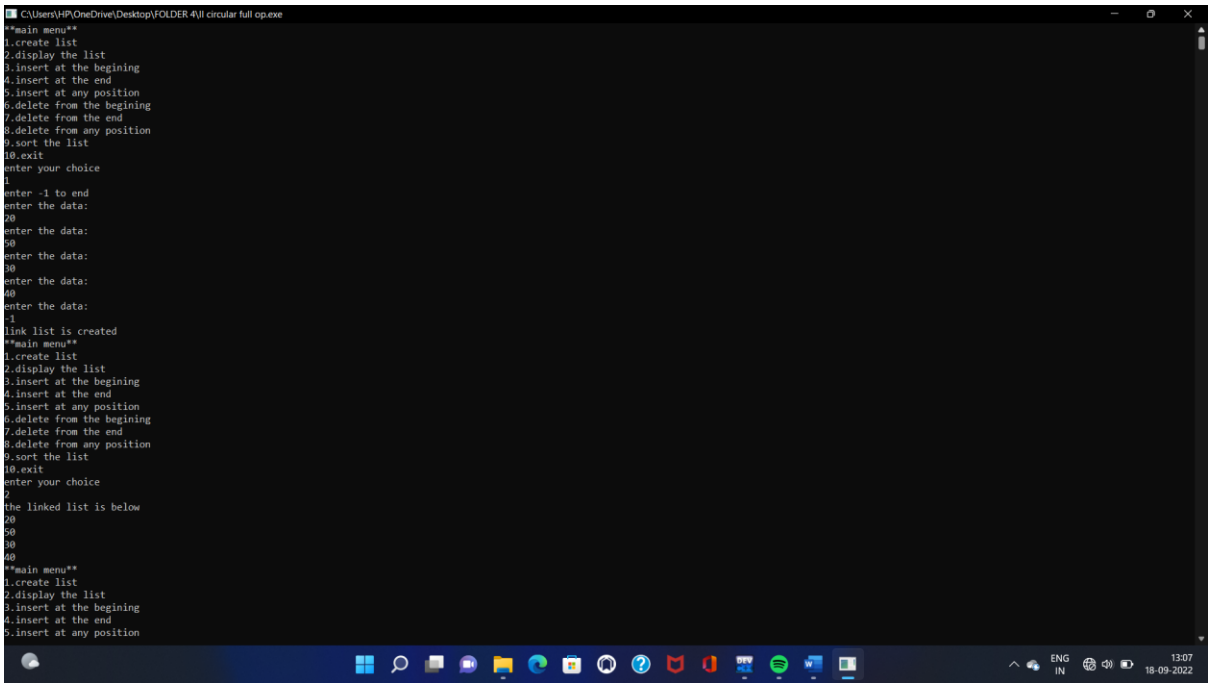
```
                printf("enter the data to be inserted: \n");
        scanf("%d",&item);
                new_node=(struct node*)malloc(sizeof(struct node*));
                new_node->data=item;
                ptr=header;
                while(ptr->link!=header)
                {
                        ptr=ptr->link;
                }
                ptr->link=new_node;
                new_node->link=header;
                printf("node inserted at the end\n");
                return header;
        }
}
struct node*insert_any(struct node*header)
{
        struct node*new_node,*ptr;
        int loc,i,item;
        if(header==NULL)
        {
                printf("overflow:insertion not possible\n");      //memory bank returns NULL
        }
        else
        {
                printf("enter the location after which the node has to be inserted\n");
        scanf("%d",&loc);
        printf("enter the data to be inserted: \n");
        scanf("%d",&item);
                new_node=(struct node*)malloc(sizeof(struct node*));
                new_node->data=item;
```

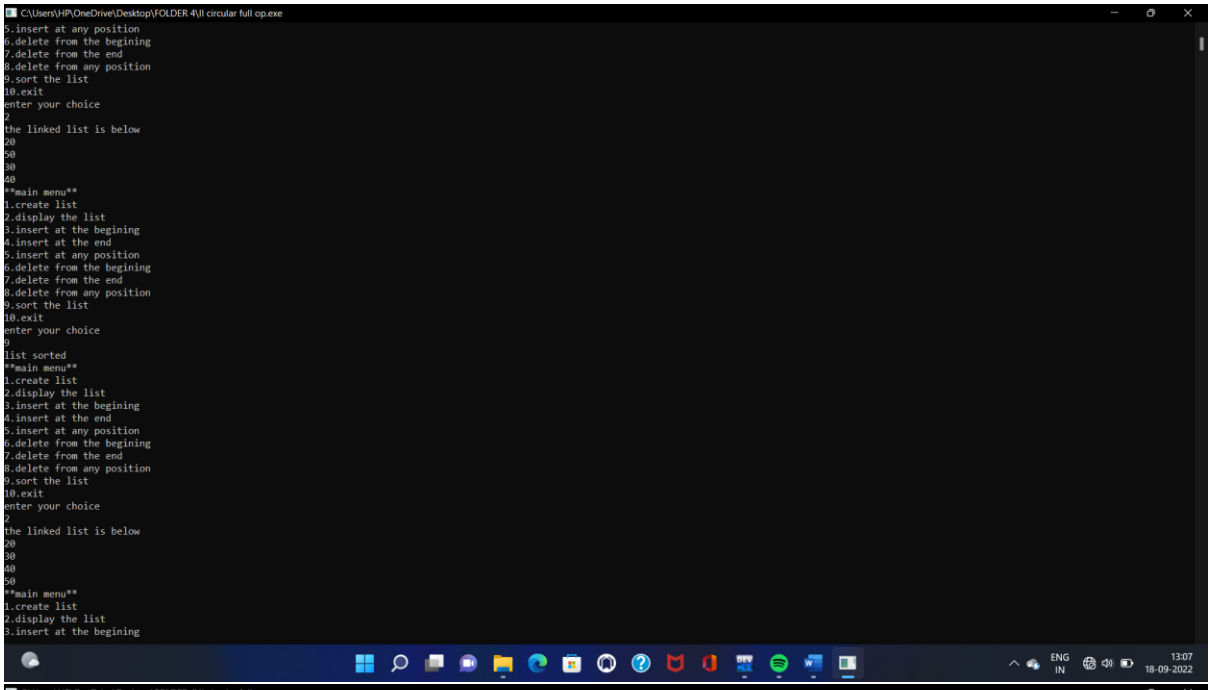
```
                ptr=header;
                for(i=0;i<loc;i++)    //the linked list is started from 0th index here
                {
                        ptr=ptr->link;
                }
                new_node->link=ptr->link;
                ptr->link=new_node;
                printf("node inserted at specific position\n");
                return header;
        }
}
struct node*delete_beg(struct node*header)
{
        struct node*ptr;
        if(header==NULL)
        {
                printf("deletion not possible\n");      //list is empty
        }
        else
        {
        ptr=header;
        while(ptr->link!=header)
        {
        ptr=ptr->link;
   }
   ptr->link=header->link;
   free(header);
   header=ptr->link;
   printf("node is deleted from the begining\n");
   return header;
        }
```

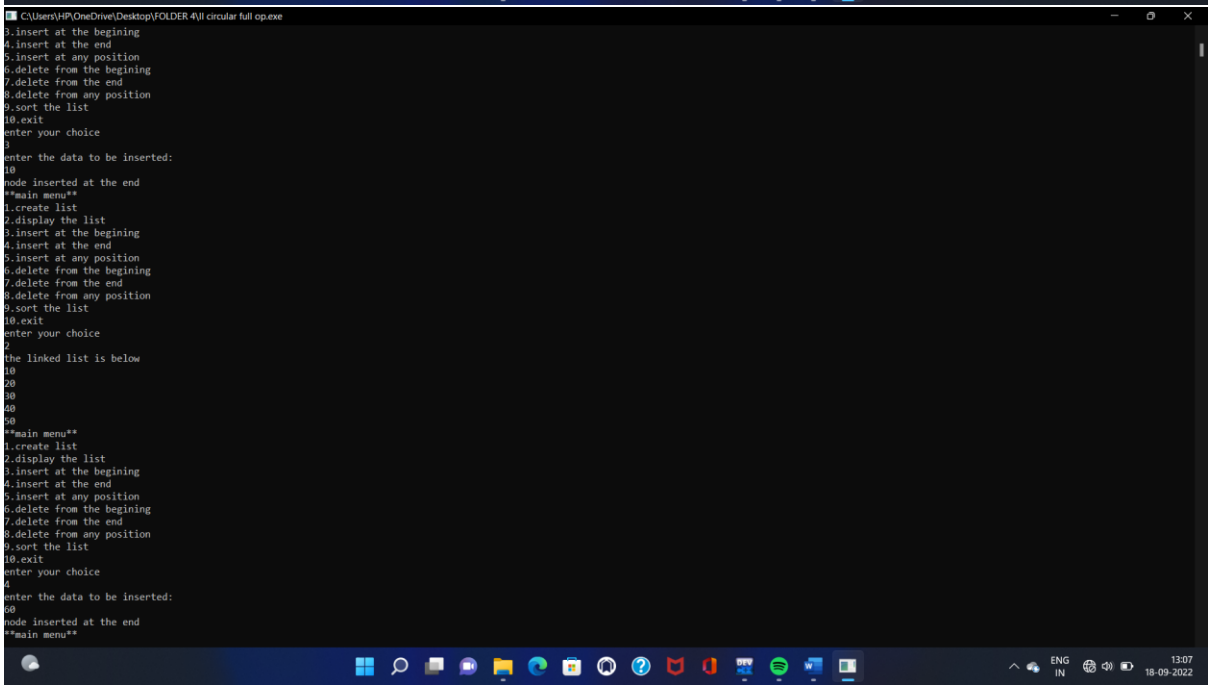
```
        /*other process to do this else part
        {
        ptr=header;
   header=header->link;
        free(ptr);
        printf("node is deleted from the begining\n");
        return header;
   }
   */
}
struct node*delete_end(struct node*header)
{
        struct node*ptr,*ptr1;
        if(header==NULL)
        {
                printf("deletion not possible\n");      //list is empty
        }
        else
        {
        ptr=header;
        while(ptr->link!=header)
        {
                ptr1=ptr;
                ptr=ptr->link;
        }
        ptr1->link=header;
        free(ptr);
        printf("node is deleted from the end\n");
        return header;
   }
}
```

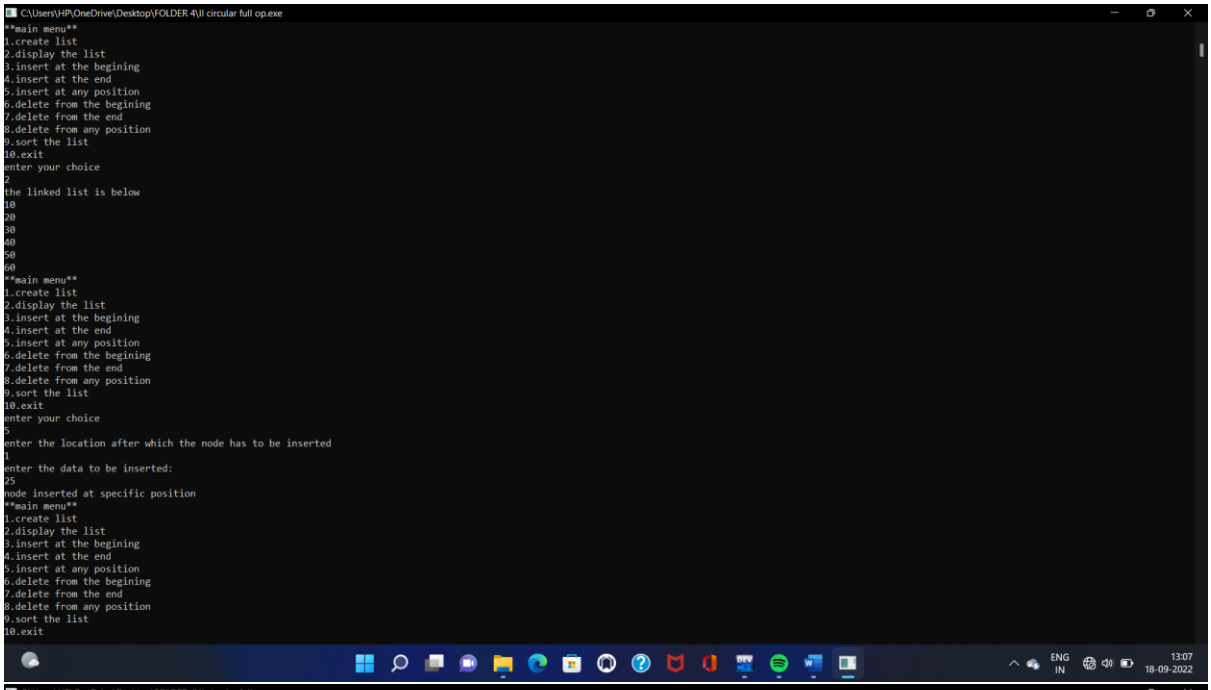
```
struct node*delete_any(struct node*header)
{
	struct node*ptr,*ptr1;
	int i,loc,item;
	if(header==NULL)
	{
		printf("deletion not possible\n");	//list is empty
	}
	else
	{
		printf("enter the location after which the node has to be deleted\n");
	   scanf("%d",&loc);
		ptr=header;
		for(i=0;i<=loc;i++)	//the linked list is started from 0th index here
	   {
		ptr1=ptr;
		ptr=ptr->link;
	   }
	ptr1->link=ptr->link;
	free(ptr);
	printf("node deleyed from specific position\n");
	return header;
	}
}
struct node*sort_list(struct node*header)
{
	struct node*ptr1,*ptr2;
	int temp;
	ptr1=header;
	while(ptr1->link!=header)
	{
```

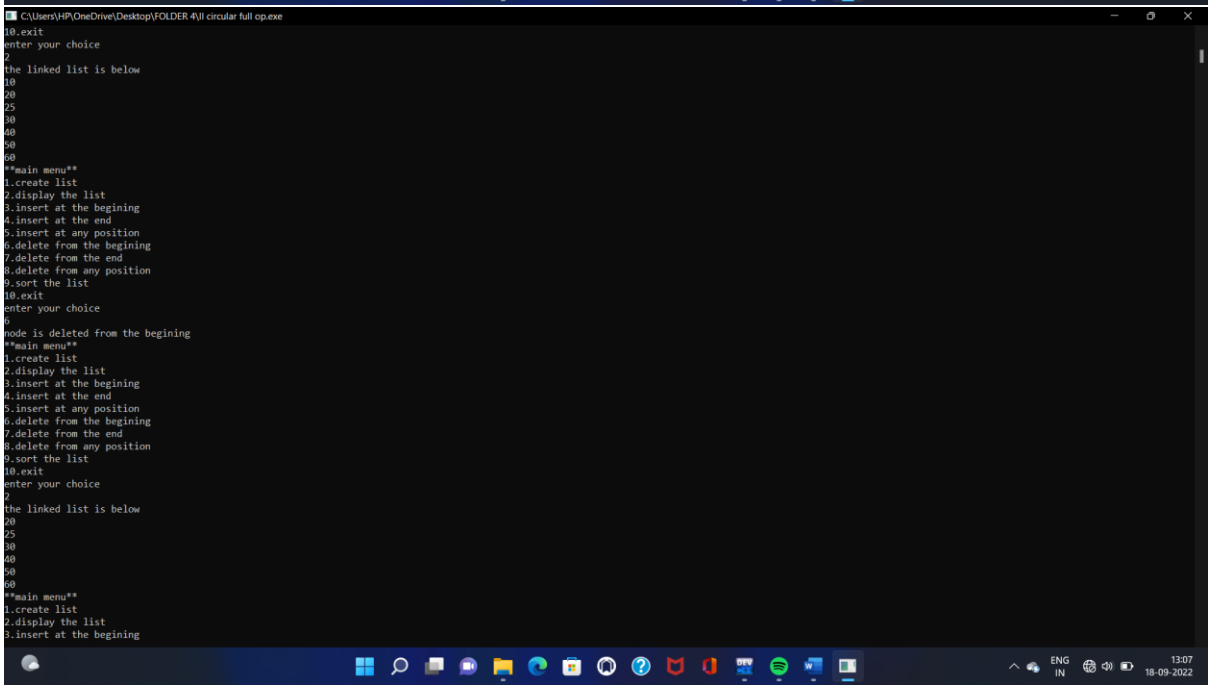
```
            ptr2=ptr1->link;
            while(ptr2!=header)      //there are atleast 2 nodes in the list
            {
                  if(ptr1->data>ptr2->data)
                  {
                        temp=ptr1->data;
                        ptr1->data=ptr2->data;
                        ptr2->data=temp;
                  }
                  ptr2=ptr2->link;
            }
            ptr1=ptr1->link;
      }
      printf("list sorted\n");
      return header;
}
```

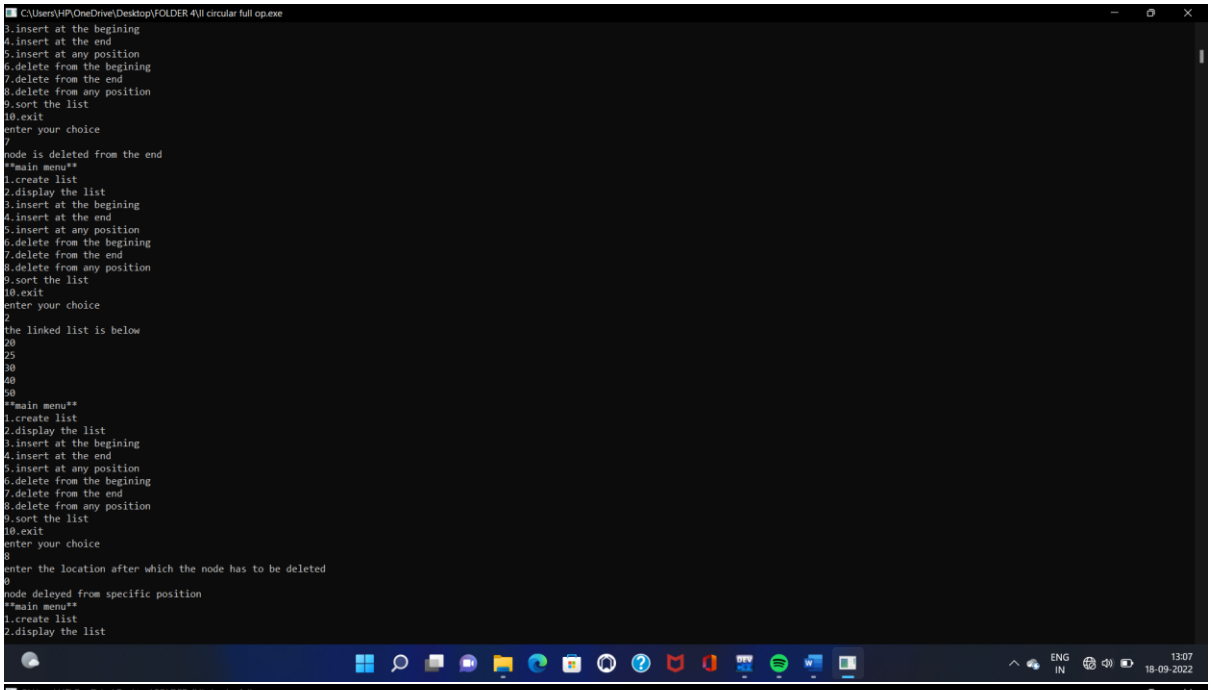


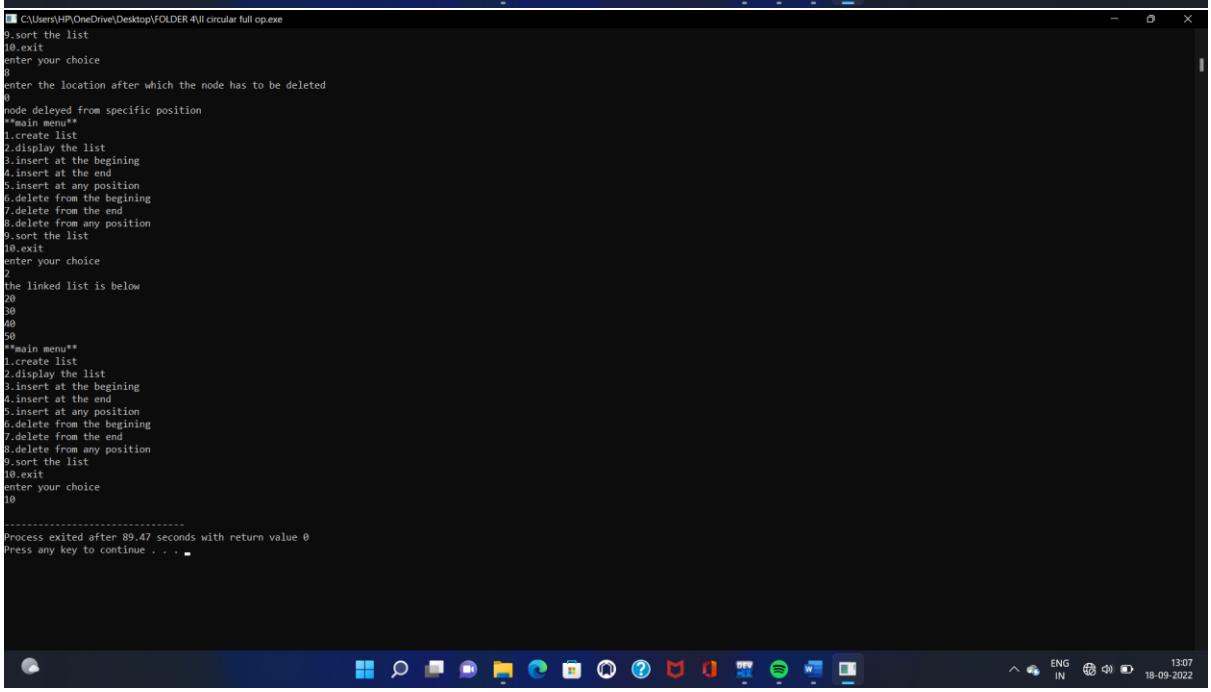

C:\Users\HP\OneDrive\Desktop\FOLDER 4\ll circular full op.exe
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
2
the linked list is below
20
50
30
40
**main menu**
1.create list
2.display the list
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
9
list sorted
**main menu**
1.create list
2.display the list
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
2
the linked list is below
20
30
40
50
**main menu**
1.create list
2.display the list
3.insert at the begining
ENG
IN
13:07
18-09-2022


C:\Users\HP\OneDrive\Desktop\FOLDER 4\ll circular full op.exe
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
3
enter the data to be inserted:
10
node inserted at the end
**main menu**
1.create list
2.display the list
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
2
the linked list is below
10
20
30
40
50
**main menu**
1.create list
2.display the list
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
4
enter the data to be inserted:
60
node inserted at the end
**main menu**
ENG
IN
13:07
18-09-2022


C:\Users\HP\OneDrive\Desktop\FOLDER 4\ll circular full op.exe
**main menu**
1.create list
2.display the list
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
2
the linked list is below
10
20
30
40
50
60
**main menu**
1.create list
2.display the list
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
5
enter the location after which the node has to be inserted
1
enter the data to be inserted:
25
node inserted at specific position
**main menu**
1.create list
2.display the list
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
ENG
IN
13:07
18-09-2022


C:\Users\HP\OneDrive\Desktop\FOLDER 4\ll circular full op.exe
10.exit
enter your choice
2
the linked list is below
10
20
25
30
40
50
60
**main menu**
1.create list
2.display the list
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
6
node is deleted from the begining
**main menu**
1.create list
2.display the list
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
2
the linked list is below
20
25
30
40
50
60
**main menu**
1.create list
2.display the list
3.insert at the begining
ENG
IN
13:07
18-09-2022


C:\Users\HP\OneDrive\Desktop\FOLDER 4\ll circular full op.exe
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
7
node is deleted from the end
**main menu**
1.create list
2.display the list
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
2
the linked list is below
20
25
30
40
50
**main menu**
1.create list
2.display the list
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
8
enter the location after which the node has to be deleted
0
node deleyed from specific position
**main menu**
1.create list
2.display the list
ENG
IN
13:07
18-09-2022


C:\Users\HP\OneDrive\Desktop\FOLDER 4\ll circular full op.exe
9.sort the list
10.exit
enter your choice
8
enter the location after which the node has to be deleted
0
node deleyed from specific position
**main menu**
1.create list
2.display the list
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
2
the linked list is below
20
30
40
50
**main menu**
1.create list
2.display the list
3.insert at the begining
4.insert at the end
5.insert at any position
6.delete from the begining
7.delete from the end
8.delete from any position
9.sort the list
10.exit
enter your choice
10
--------------------------------
Process exited after 89.47 seconds with return value 0
Press any key to continue . . .
ENG
IN
13:07
18-09-2022