/*linked list random deletion and insertion by value*/

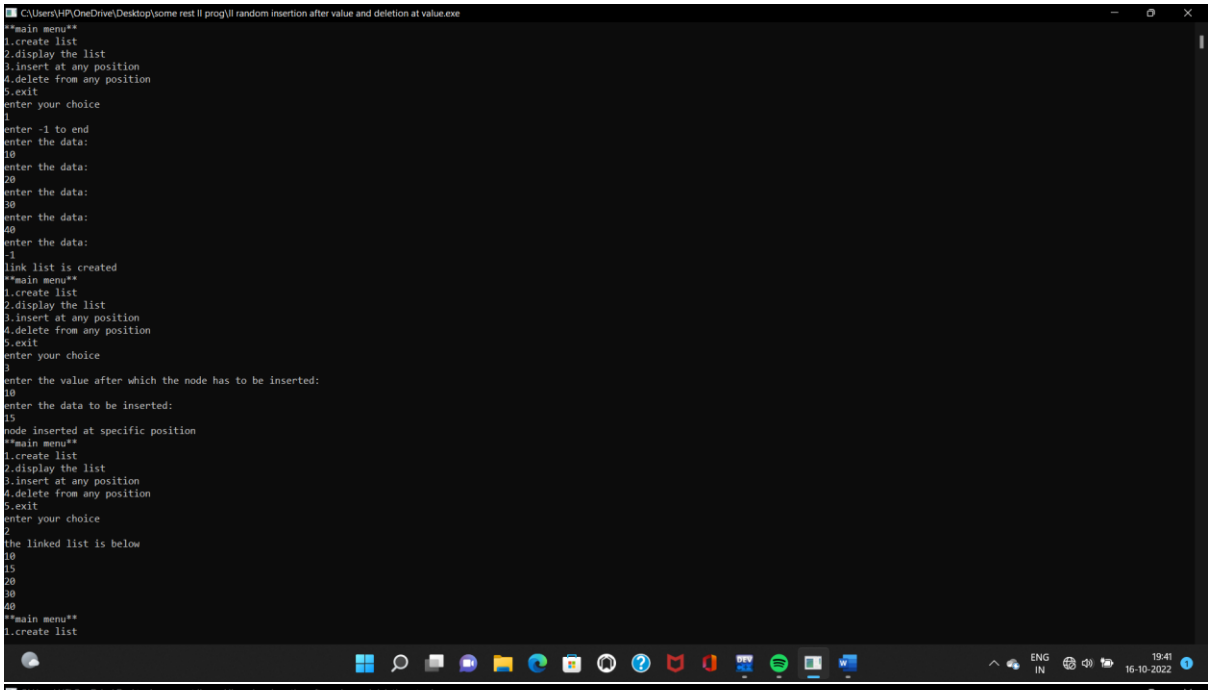
```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node*link;
};
struct node*header;
struct node*create_ll(struct node*);
struct node*display(struct node*);
struct node*insert_any(struct node*);
struct node*delete_any(struct node*);
int main()
{
int choice=0;
while(choice!=5)
{
printf("**main menu**\n");
printf("1.create list\n2.display the list\n3.insert at any position\n4.delete from any
position\n5.exit\n");
printf("enter your choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:header=create_ll(header);
break;
case 2:header=display(header);
break;
case 3:header=insert_any(header);
break;
```

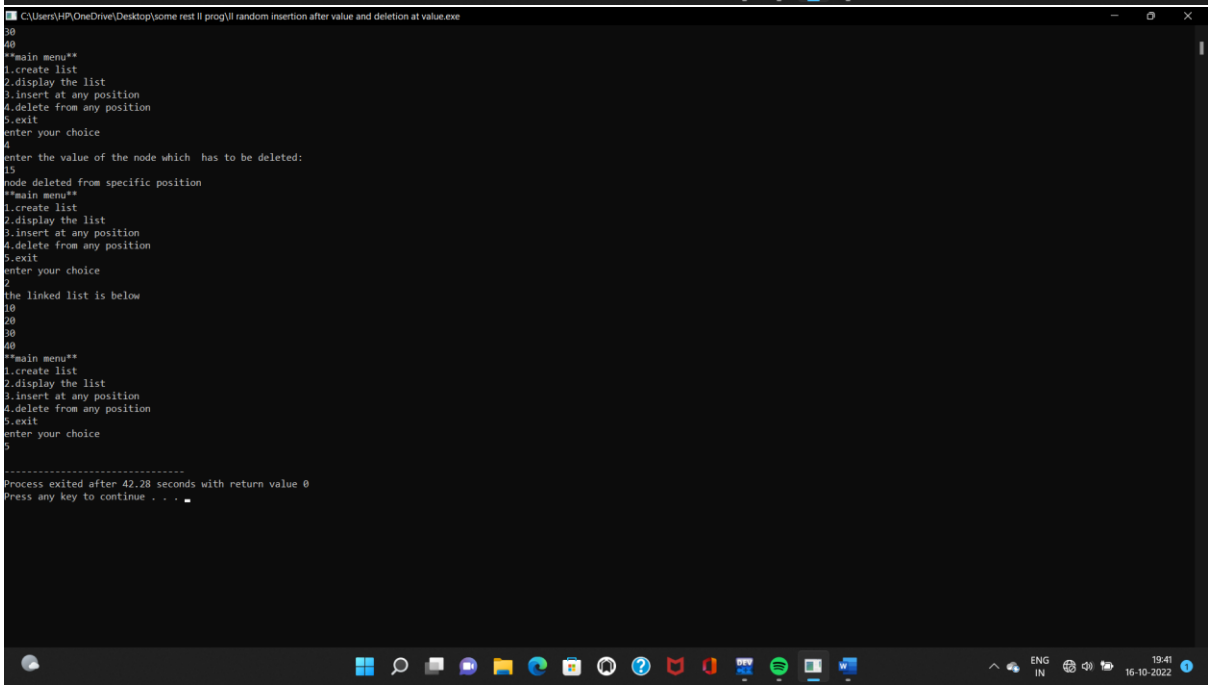
```
case 4:delete_any(header);
break;
case 5:exit(0);
default:
printf("invalid choice\n");
}
}
}
struct node*create_ll(struct node*header)
{
struct node*new_node,*ptr;
int item;
printf("enter -1 to end\n");
printf("enter the data: \n");
scanf("%d",&item);
while(item!=-1)
{
new_node=(struct node*)malloc(sizeof(struct node*));
new_node->data=item;
if(header==NULL) //list is empty
{
new_node->link=NULL;
header=new_node;
}
else
{
ptr=header;
while(ptr->link!=NULL)
{
ptr=ptr->link;
 }
```

```
ptr->link=new_node;
new_node->link=NULL;
}
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!=NULL) //list is not empty
{
printf("%d\n",ptr->data);
ptr=ptr->link;
 }
 return header;
}
struct node*insert_any(struct node*header)
{
struct node*new_node,*ptr;
int val,item;
if(header==NULL)
{
printf("overflow:insertion not possible\n"); //memory bank returns NULL
}
else
{
```

```
printf("enter the value after which the node has to be inserted: \n");
 scanf("%d",&val);
 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->data!=val)
{
ptr=ptr->link;
}
new_node->link=ptr->link;
ptr->link=new_node;
printf("node inserted at specific position\n");
return header;
}
}
struct node*delete_any(struct node*header)
{
struct node*ptr,*ptr1;
int val;
if(header==NULL)
{
printf("deletion not possible\n"); //list is empty
}
else
{
printf("enter the value of the node which  has to be deleted: \n");
 scanf("%d",&val);
ptr=header;
while(ptr->data!=val) //when there is no ptr1,then its' data(NULL) obviously not equal to val
```

```
 {
ptr1=ptr;
ptr=ptr->link;
 }
ptr1->link=ptr->link;
free(ptr);
printf("node deleted from specific position\n");
return header;
}
}
```


C:\Users\HP\OneDrive\Desktop\some rest II prog\II random insertion after value and deletion at value.exe
main menu
1.create list
2.display the list
3.insert at any position
4.delete from any position
5.exit
enter your choice
1
enter -1 to end
enter the data:
10
enter the data:
20
enter the data:
30
enter the data:
40
enter the data:
-1
link list is created
main menu
1.create list
2.display the list
3.insert at any position
4.delete from any position
5.exit
enter your choice
3
enter the value after which the node has to be inserted:
10
enter the data to be inserted:
15
node inserted at specific position
main menu
1.create list
2.display the list
3.insert at any position
4.delete from any position
5.exit
enter your choice
2
the linked list is below
10
15
20
30
40
main menu
1.create list


C:\Users\HP\OneDrive\Desktop\some rest II prog\II random insertion after value and deletion at value.exe
30
40
main menu
1.create list
2.display the list
3.insert at any position
4.delete from any position
5.exit
enter your choice
4
enter the value of the node which has to be deleted:
15
node deleted from specific position
main menu
1.create list
2.display the list
3.insert at any position
4.delete from any position
5.exit
enter your choice
2
the linked list is below
10
20
30
40
main menu
1.create list
2.display the list
3.insert at any position
4.delete from any position
5.exit
enter your choice
5

Process exited after 42.28 seconds with return value 0
Press any key to continue . . .