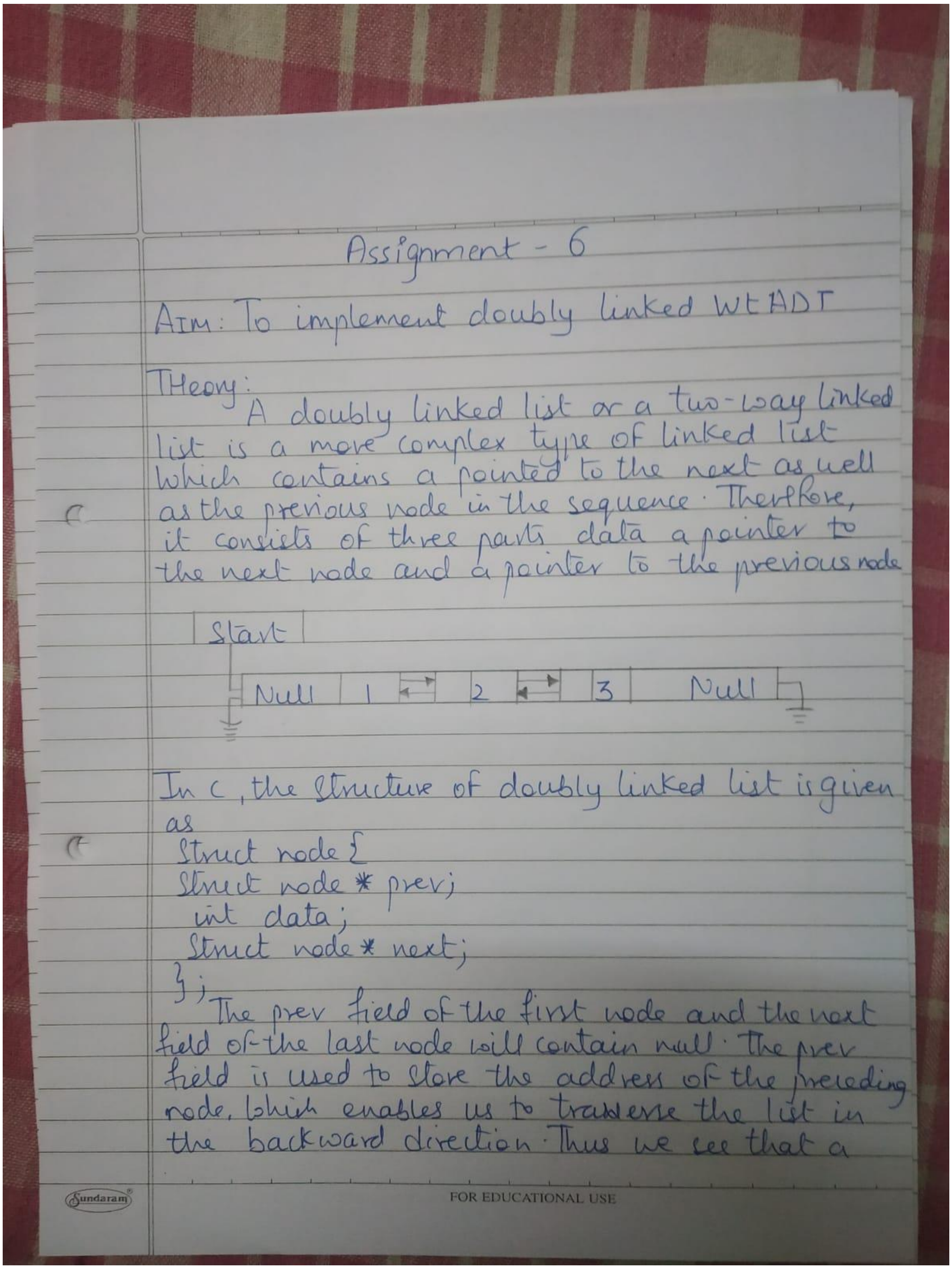
COMPUTER ENGINEERING

DS ODD SEM 2021-22/EXPERIMENT 6

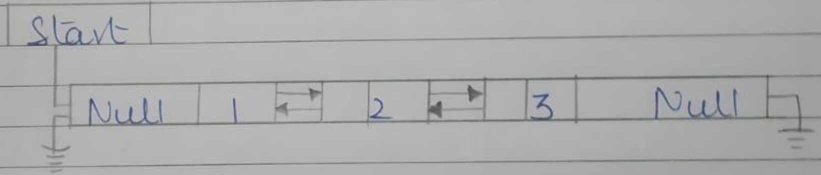
NAME:- GAURAV AMARNANI (D7A, 67)



Assignment - 6

AIM: To implement doubly linked WtADT

THeory:
A doubly linked list or a two-way linked list is a more complex type of linked list which contains a pointed to the next as well as the previous node in the sequence. Therefore, it consists of three parts data a pointer to the next node and a pointer to the previous node

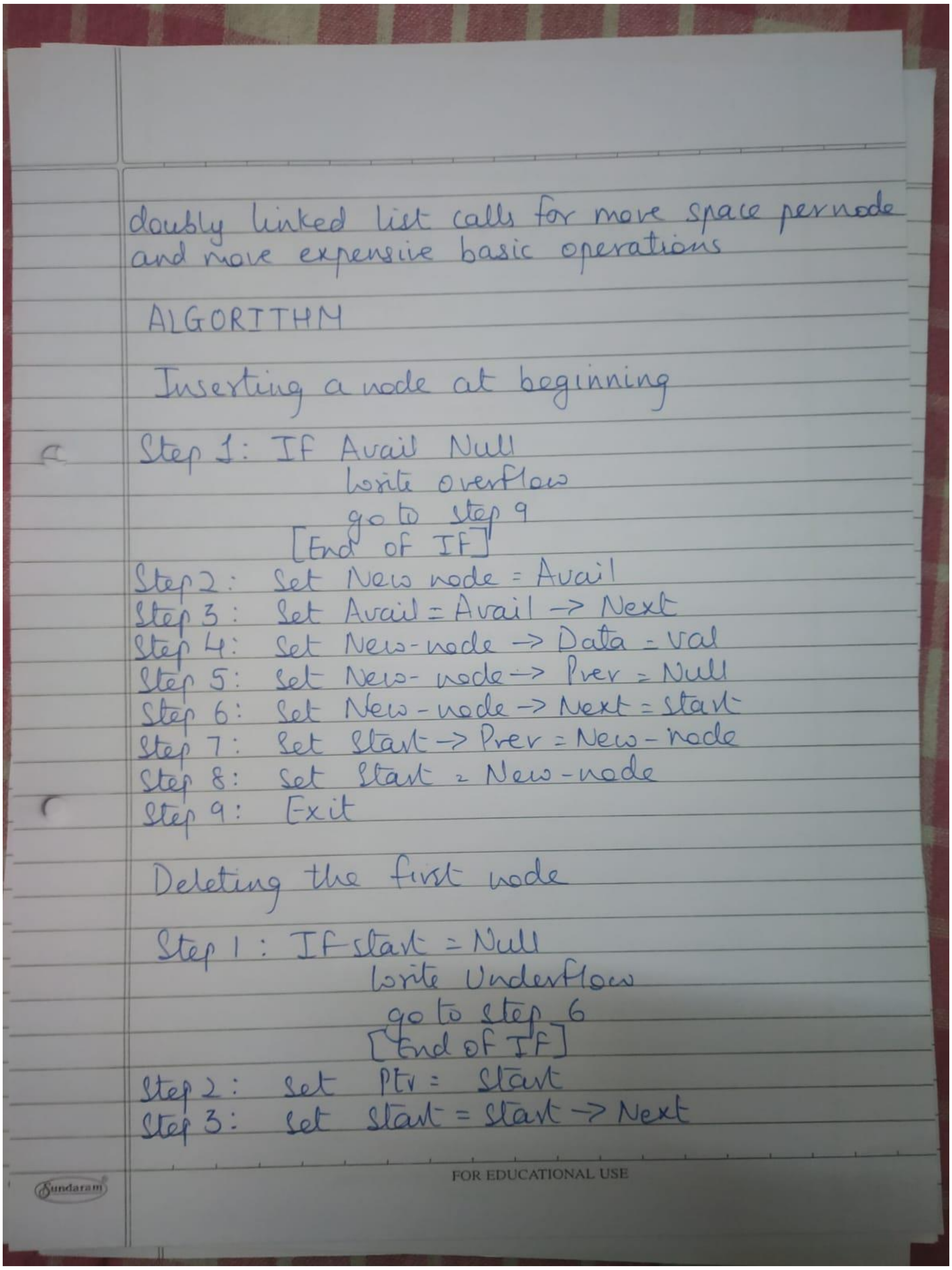


In c, the structure of doubly linked list is given as

```
Struct node{
Struct node * prev;
 int data;
 Struct node * next;
};
```

The prev field of the first node and the next field of the last node will contain null. The prev field is used to store the address of the preceding node, which enables us to traverse the list in the backward direction. Thus we see that a

FOR EDUCATIONAL USE



doubly linked list calls for more space per node and more expensive basic operations

ALGORITHM

Inserting a node at beginning

```
Step 1: IF Avail Null
                Write overflow
                go to step 9
        [End of IF]
Step 2:  Set New node = Avail
Step 3:  Set Avail = Avail -> Next
Step 4:  Set New-node -> Data = Val
Step 5:  Set New- node -> Prev = Null
Step 6:  Set New - node -> Next = start
Step 7:  Set Start -> Prev = New - node
Step 8:  Set Start = New-node
Step 9:  Exit
```

Deleting the first node

```
Step 1 : IF start = Null
                Write Underflow
                go to step 6
         [End of IF]
Step 2:  set  Ptr = Start
Step 3:  set  Start = Start -> Next
```

FOR EDUCATIONAL USE

```
Step 4: Set Start → Prev = Null
Step 5: free PTR
Step 6: Exit
```

Conclusion:

By Performing this experiment we understood the concept of doubly linked list its applications and how we can implement it in our program.

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
struct node {
struct node *prev;
struct node *next;
int data;
};
struct node *head;
void insertion_beginning();
void insertion_last();
void insertion_specified();
void deletion_beginning();
void deletion_last();

void deletion_specified();
void display();
void search();
void main () {
int choice =0;
while(choice != 9) {
printf("\nChoose one option from the following list ...\n");
printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete from
Beginning 5.Delete from last\n 6.Exit\n");
printf("\nEnter your choice?\n");
scanf("\n%d",&choice);
switch(choice) {
case 1:
insertion_beginning();
break;
case 2:
insertion_last();
break;
case 3:
insertion_specified();
break;
case 4:
deletion_beginning();
break;
case 5:
deletion_last();
break;
case 6:
exit(0);
default:
printf("Please enter valid choice..");
}
}
}
void insertion_beginning() {
struct node *ptr;
int item;
ptr = (struct node *)malloc(sizeof(struct node));
```

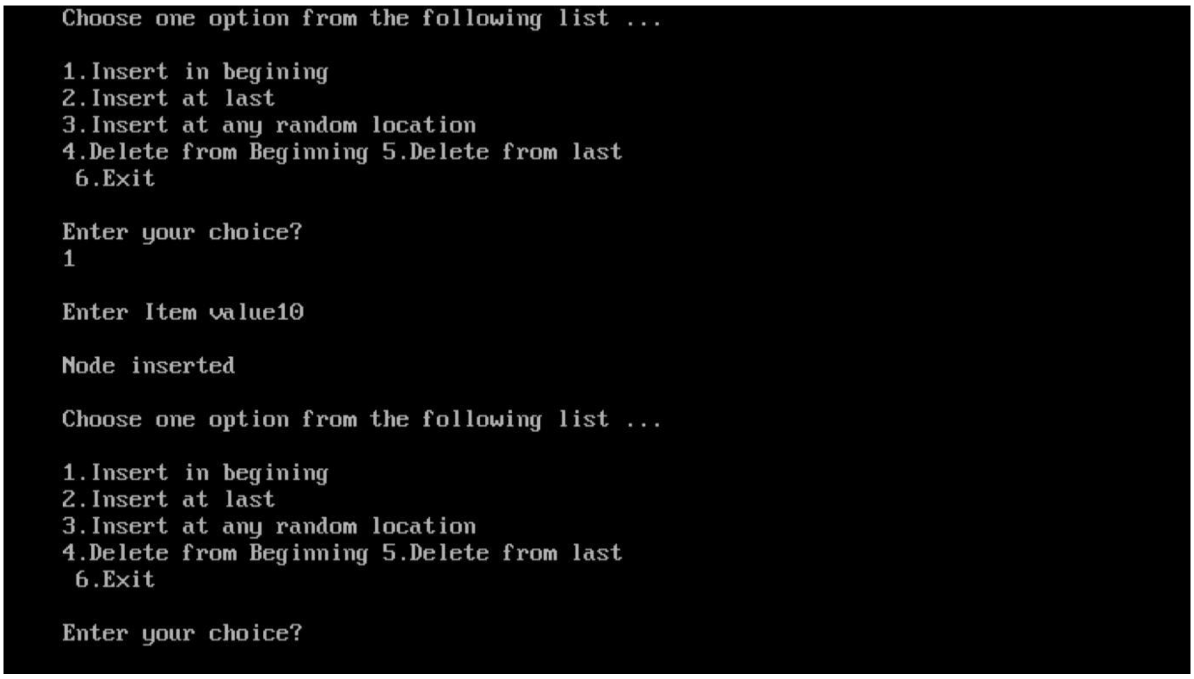
```
if(ptr == NULL) {
printf("\nOVERFLOW");
}
else {
printf("\nEnter Item value");
scanf("%d",&item);
if(head==NULL) {
ptr->next = NULL;
ptr->prev=NULL;
ptr->data=item;
head=ptr;
}
else {
ptr->data=item;
ptr->prev=NULL;
ptr->next = head;
head->prev=ptr;
head=ptr;
}
printf("\nNode inserted\n");
}
}
void insertion_last() {
struct node *ptr,*temp;
int item;
ptr = (struct node *) malloc(sizeof(struct node));
if(ptr == NULL) {
printf("\nOVERFLOW");
}
else {
printf("\nEnter value");
scanf("%d",&item);
ptr->data=item;
if(head == NULL) {
ptr->next = NULL;
ptr->prev = NULL;
head = ptr;
}
else {
temp = head;
while(temp->next!=NULL) {
temp = temp->next;
}
temp->next = ptr;
ptr ->prev=temp;
ptr->next = NULL;
}
}
printf("\nnode inserted\n");
}
void insertion_specified() {
struct node *ptr,*temp;
int item,loc,i;
ptr = (struct node *)malloc(sizeof(struct node));
if(ptr == NULL) {
```

```
printf("\n OVERFLOW");
}
else {
temp=head;
printf("Enter the location");
scanf("%d",&loc);
for(i=0;i<loc;i++) {
temp = temp->next;
if(temp == NULL) {
printf("\n There are less than %d elements", loc);
return;
}
}
printf("Enter value");
scanf("%d",&item);
ptr->data = item;
ptr->next = temp->next;
ptr -> prev = temp;
temp->next = ptr;
temp->next->prev=ptr;
printf("\nnode inserted\n");
}
}
void deletion_beginning() {
struct node *ptr;
if(head == NULL) {
printf("\n UNDERFLOW");
}
else if(head->next == NULL) {
head = NULL;
free(head);
printf("\nnode deleted\n");
}
else {
ptr = head;
head = head -> next;
head -> prev = NULL;
free(ptr);
printf("\nnode deleted\n");
}
}
void deletion_last() {
struct node *ptr;
if(head == NULL) {
printf("\n UNDERFLOW");
}
else if(head->next == NULL) {
head = NULL;
free(head);
printf("\nnode deleted\n");
}
else {
ptr = head;
if(ptr->next != NULL) {
ptr = ptr -> next;
```

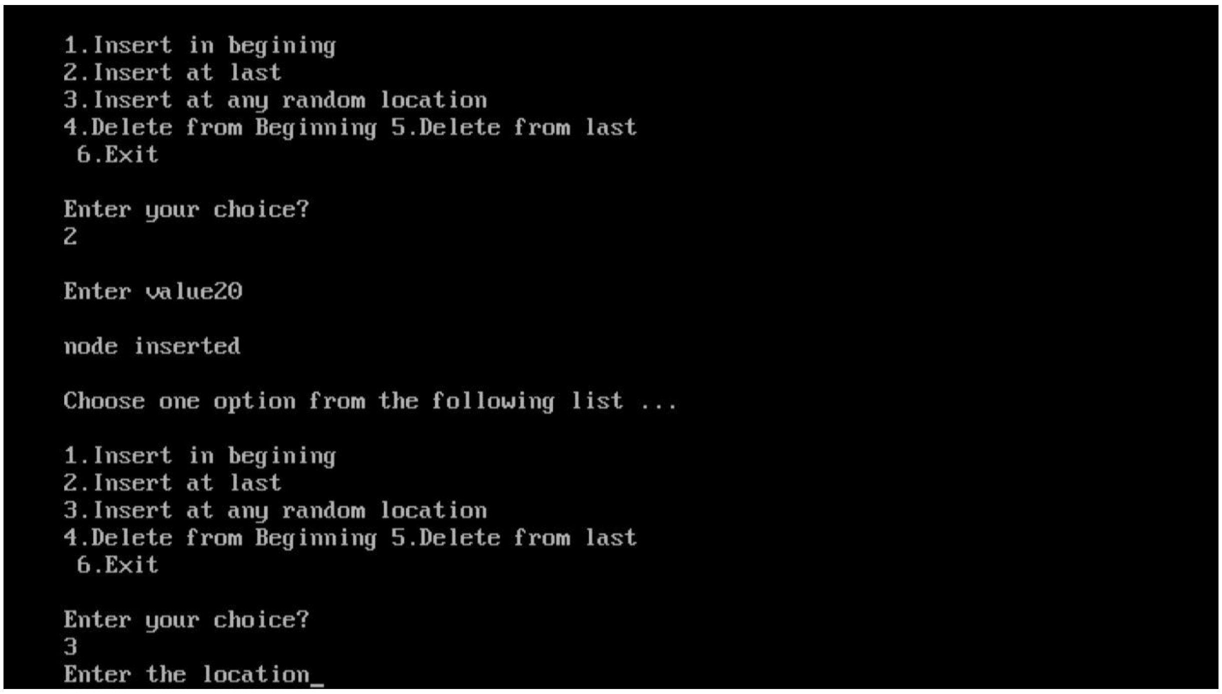
```
}
ptr -> prev -> next = NULL;
free(ptr);
printf("\nnode deleted\n");
}
}
void deletion_specified() {
struct node *ptr, *temp;
int val;
printf("\n Enter the data after which the node is to be deleted : ");
scanf("%d", &val);
ptr = head;
while(ptr -> data != val) ptr = ptr -> next;
if(ptr -> next == NULL) {
printf("\nCan't delete\n");
}
else if(ptr -> next -> next == NULL) {
ptr ->next = NULL;
}
else {
temp = ptr -> next;
ptr -> next = temp -> next;
temp -> next -> prev = ptr;
free(temp);
printf("\nnode deleted\n");
}
}
void display() {
struct node *ptr;
printf("\n printing values...\n");
ptr = head;
while(ptr != NULL) {
printf("%d\n",ptr->data);
ptr=ptr->next;
}
}
void search() {
struct node *ptr;
int item,i=0,flag;
ptr = head;
if(ptr == NULL) {
printf("\nEmpty List\n");
}
else {
printf("\nEnter item which you want to search?\n");
scanf("%d",&item);
while (ptr!=NULL) {
if(ptr->data == item) {
printf("\nitem found at location %d ",i+1);
flag=0;
break;
}
else {
flag=1;
}
```

```
i++;
ptr = ptr -> next;
}
if(flag==1) {
printf("\nItem not found\n");
}
}
}
```

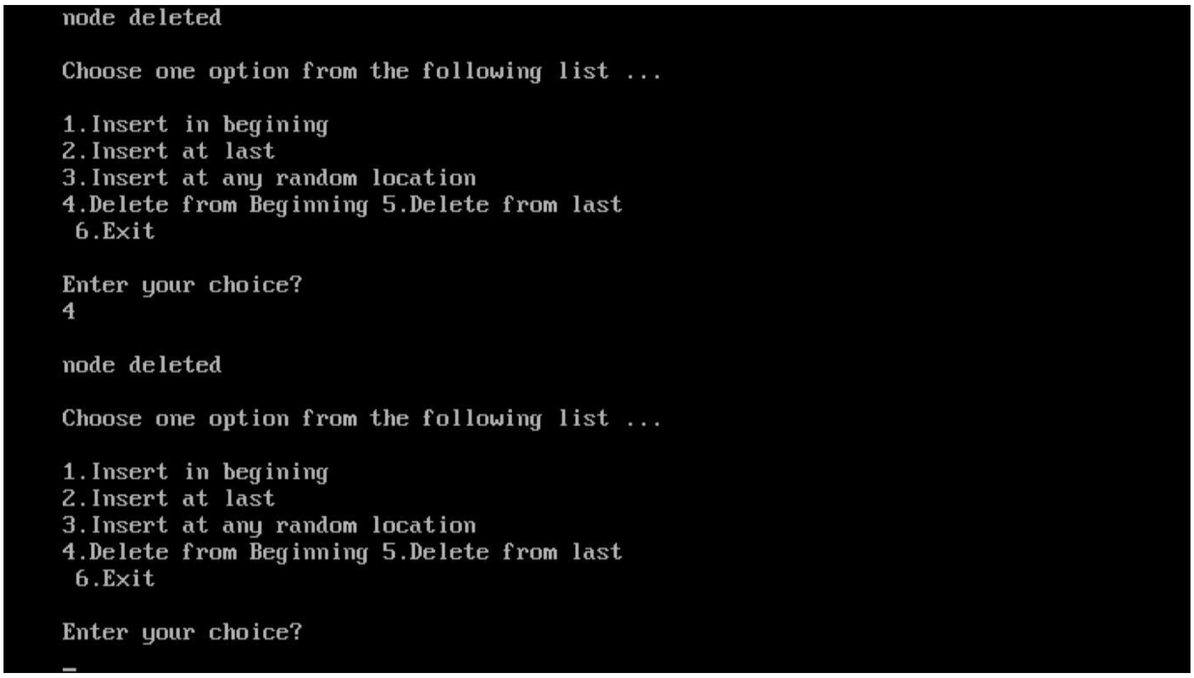
OUTPUT:-



```
Choose one option from the following list ...

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning 5.Delete from last
 6.Exit

Enter your choice?
1

Enter Item value10

Node inserted

Choose one option from the following list ...

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning 5.Delete from last
 6.Exit

Enter your choice?
```



```
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning 5.Delete from last
 6.Exit

Enter your choice?
2

Enter value20

node inserted

Choose one option from the following list ...

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning 5.Delete from last
 6.Exit

Enter your choice?
3
Enter the location_
```



```
node deleted

Choose one option from the following list ...

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning 5.Delete from last
 6.Exit

Enter your choice?
4

node deleted

Choose one option from the following list ...

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning 5.Delete from last
 6.Exit

Enter your choice?
_
```