Program3: Doubly Linked List Operations

```
#include<stdio.h>
#include<alloc.h>
#include<conio.h>
typedef struct NodeType
{
int info;
struct NodeType *next,*prev;
}Node;
void main()
{
Node *head,*tail,*toadd,*todel,*tocount, *tosearch, *todisp;
int no,i,data;
clrscr();
head=tail=(Node *)NULL;
printf("\nCreate a doubly linked list by adding number of nodes, how many? ");
scanf("%i",&no);
for(i=1;i<=no;i++)
{
toadd=(Node *)malloc(sizeof(Node));
if(toadd==(Node *)NULL)
{
printf("\nMemory allocation error! node not created in memory");
return;
}
printf("\nEnter the data for node information:\t");
scanf("%i",&data);
toadd->info=data;
if(head==(Node *)NULL)
```

```
{
toadd->prev=toadd->next=(Node*)NULL;
head=tail=toadd;
}
else
{
toadd->prev=tail;
toadd->next=(Node *)NULL;
toadd->prev->next=toadd;
tail=toadd;
}
}
toadd=(Node *)malloc(sizeof(Node));
if(toadd==(Node*)NULL)
{
printf("\nMemory not allocated for new node");
return;
}
printf("\nEnter the data to add at top of DLL: ");
scanf("%i",&data);
toadd->info=data;
toadd->prev=(Node*)NULL;
toadd->next=(Node*)head;
head->prev=toadd;
head=toadd;
toadd=(Node*)malloc(sizeof(Node));
if(toadd==(Node*)NULL)
{
printf("\nMemory not allocated for new node");
```

```
return;
}
printf("\nEnter the data to add at end of DLL:\t");
scanf("%i",&data);
toadd->info=data;
toadd->next=(Node*)NULL;
toadd->prev=tail;
tail->next=toadd;
tail=toadd;
int item;
printf("\nEnter the item to search in DLL:\t");
scanf("%i",&item);
tosearch=head;
while(tosearch->next!=(Node*)NULL)
{
if(tosearch->info==item)
{
toadd=(Node*)malloc(sizeof(Node));
if(toadd==(Node*)NULL)
{
printf("\nMemory allocation error");
break;
}
printf("\nAdding data after any node:\t");
scanf("%i",&data);
toadd->info=data;
toadd->prev=tosearch;
toadd->next=tosearch->next;
tosearch->next=toadd;
```

```
toadd->next->prev=toadd;
break;
}
tosearch=tosearch->next;
}
if(tosearch->next==(Node*)NULL)
{
toadd=(Node*)malloc(sizeof(Node));
if(toadd==(Node*)NULL)
{
printf("\nMemory allocation error");
return;
}
printf("\nAdding data to add after any node:\t");
scanf("%i",&data);
toadd->info=data;
toadd->prev=tail;
toadd->next=(Node*)NULL;
tail->next=toadd;
tail=toadd;
}

printf("\nElements in input order\n");
todisp=head;
while(todisp!=(Node *)NULL)
{
printf("\n%d",todisp->info);
todisp=todisp->next;
}
```

```
printf("\nElements in reverse input order\n");
todisp=tail;
while(todisp!=(Node *)NULL)
{
printf("\n%d",todisp->info);
todisp=todisp->prev;
}
tocount=head;
i=0;
while(tocount!=(Node*)NULL)
{
i++;
tocount=tocount->next;
}
printf("\nTotal %d nodes present now",i);
tosearch=tail;
printf("\nEnter the data to search in DLL: ");
scanf("%i",&data);
while(tosearch!=(Node*)NULL)
{
if(tosearch->info==data)
break;
tosearch=tosearch->prev;
}
if(tosearch==(Node*)NULL)
printf("\n%d data not found in dll",data);
else
printf("\n%d is found",data);
todel=head;
```

```
head=todel->next;
todel->next->prev=(Node*)NULL;
printf("\n%d at top of DLL is deleted now",todel->info);
free(todel);
todel=tail;
tail=todel->prev;
todel->prev->next=(Node*)NULL;
printf("\n%d is being deleted now from tail of DLL",todel->info);
free(todel);
todel=head;
while(todel!=(Node *)NULL)
{
printf("\nNode of %d information is being deleted",todel->info);
head=todel->next;
free(todel);
todel=head;
}
head=tail=(Node *)NULL;
}
```