DS LAB-PROG 9-DOUBLY LINKED LIST

Program and output

Mallika Prasad

1BM19CS081

9.12.2020

Program 9-

WAP to Implement doubly link list with primitive operations a) Create a doubly linked list. b) Insert a new node to the left of the node. c) Delete the node based on a specific value d) Display the contents of the list e) Delete the duplicates [plus other functions]

```
#include<stdio.h>

#include<stdlib.h>


struct node

{

        int info;

        struct node *llink;

        struct node *rlink;

        };

typedef struct node *NODE;

NODE getnode()

{

        NODE x;

        x=(NODE)malloc(sizeof(struct node));

        if(x==NULL)
```

```
        {
                printf("mem full\n");
                exit(0);
                }
        return x;
        }
void freenode(NODE x)
{
        free(x);
}

NODE dinsert_front(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
cur=head->rlink;
head->rlink=temp;
temp->llink=head;
temp->rlink=cur;
cur->llink=temp;
return head;
}
```

```
NODE dinsert_rear(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
cur=head->llink;
head->llink=temp;
temp->rlink=head;
temp->llink=cur;
cur->rlink=temp;
return head;
}

NODE ddelete_front(NODE head)
{
NODE cur,next;
if(head->rlink==head)
{
printf("list empty\n");
return head;
}
cur=head->rlink;
next=cur->rlink;
head->rlink=next;
```

```
next->llink=head;
printf("the node deleted is %d\n",cur->info);
freenode(cur);
return head;
}

NODE ddelete_rear(NODE head)
{
NODE cur,prev;
if(head->rlink==head)
{
printf("list empty\n");
return head;
}
cur=head->llink;
prev=cur->llink;
head->llink=prev;
prev->rlink=head;
printf("the node deleted is %d\n",cur->info);
freenode(cur);
return head;
}

void display(NODE head)
```

```
{
NODE temp;
if(head->rlink==head)
{
printf("list empty\n");
return;
}
printf("contents of list\n");
temp=head->rlink;
while(temp!=head)
{
printf("%d\n",temp->info);
temp=temp->rlink;
}
printf("\n");
}

NODE insert_leftpos(int item,NODE head)
{
NODE temp,cur,prev;
if(head->rlink==head)
{
printf("list empty\n");
return head;
```

```
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)break;
cur=cur->rlink;
}
if(cur==head)
{
 printf("key not found\n");
 return head;
 }
 prev=cur->llink;
 printf("enter towards left of %d=",item);
 temp=getnode();
 scanf("%d",&temp->info);
 prev->rlink=temp;
 temp->llink=prev;
 cur->llink=temp;
 temp->rlink=cur;
 return head;
}

NODE insert_rightpos(int item,NODE head)
```

```
{
NODE temp,cur,next;
if(head->rlink==head)
{
printf("list empty\n");
return head;
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)break;
cur=cur->rlink;
}
if(cur==head)
{
 printf("key not found\n");
 return head;
 }
 next=cur->rlink;
 printf("enter towards right of %d=",item);
 temp=getnode();
 scanf("%d",&temp->info);
 next->llink=temp;
 temp->rlink=next;
```

```
 cur->rlink=temp;
 temp->llink=cur;
 return head;
}

NODE delete_dup_key(int item,NODE head)
{
NODE prev, cur, next;
   int count;
   if (head->rlink == head)
   {
      printf("List is Empty!");
      return head;
   }
   count = 0;
   cur = head->rlink;
   while (cur != head)
   {
      if (item != cur->info)
         cur = cur->rlink;
      else
      {
         count++;
         if(count==1)
```

```
        {
            cur = cur->rlink;
        }
        if(count!=1)
        {
            prev = cur->llink;
            next = cur->rlink;
            prev->rlink = next;
            next->llink = prev;
            freenode(cur);
            cur = next;
        }

      }
   }
   if (count == 0)
      printf("Key not found");
   else
      printf("Duplicates are deleted\n");

   return head;
}

void search(int item,NODE head){
```

```
    NODE cur;
    if(head->rlink==head)
    {
     printf("List Empty");
     return ;
     }
     cur=head->rlink;
     while(cur!=head)
    {
    if(item==cur->info)break;
    cur=cur->rlink;
    }
    if(cur==head)
    {printf("search unsuccessfull\n");
       return;
    }
    printf("search successful\n");

}

NODE delete_value(int item,NODE head)
{
NODE prev,cur,next;
int count;
```

```
   if(head->rlink==head)
    {
     printf("List Empty");
     return head;
     }
count=0;
cur=head->rlink;
while(cur!=head)
{
  if(item!=cur->info)
  cur=cur->rlink;
  else
 {
  count++;
  prev=cur->llink;
  next=cur->rlink;
  prev->rlink=next;
  next->llink=prev;
  freenode(cur);
 }
}
if(count==0)
  printf("key not found\n");
  else
```

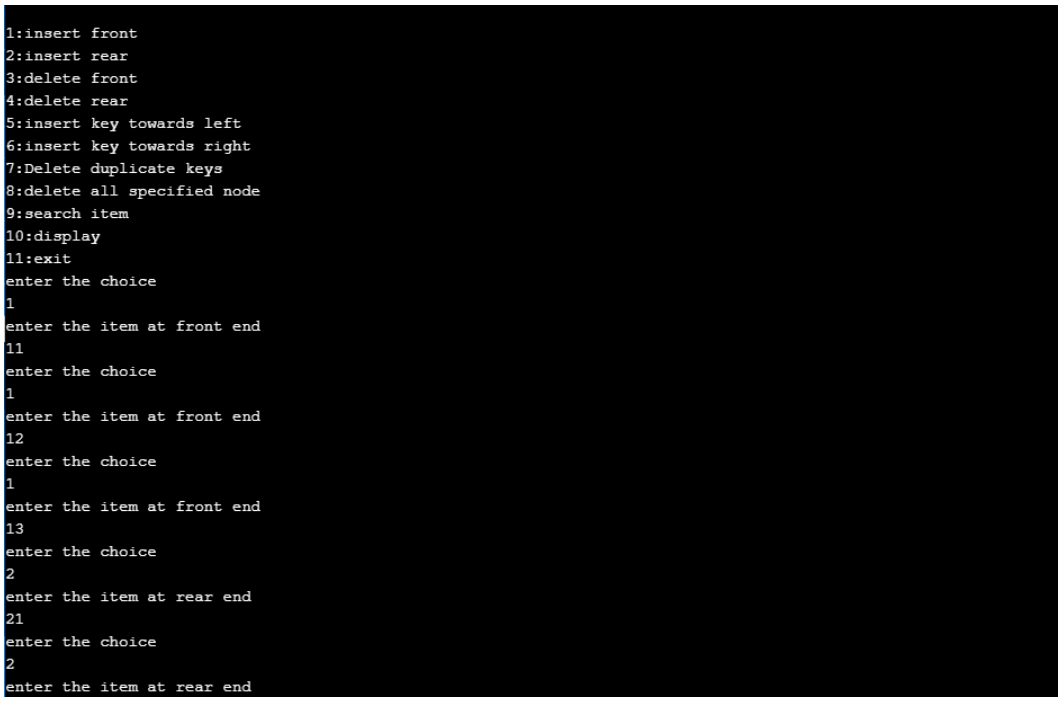
```
 printf("key found at %d positions and are deleted\n",count);
return head;
}

int main()
{
NODE head,last;
int item, choice;
head=getnode();
head->rlink=head;
head->llink=head;
        printf("\n1:insert front\n2:insert rear\n3:delete front\n4:delete rear\n5:insert key towards left\n6:insert key towards right\n7:Delete duplicate keys\n8:delete all specified node\n9:search item\n10:display\n11:exit\n");
        do
        {
        printf("enter the choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
                case 1: printf("enter the item at front end\n");
                        scanf("%d",&item);
                        last=dinsert_front(item,head);
                        break;
                case 2: printf("enter the item at rear end\n");
```

```
                scanf("%d",&item);
                last=dinsert_rear(item,head);
                break;
case 3:last=ddelete_front(head);
                break;
case 4: last=ddelete_rear(head);
                break;
case 5: printf("enter the key item\n");
        scanf("%d",&item);
        last=insert_leftpos(item,head);
        break;
case 6: printf("enter the key item\n");
        scanf("%d",&item);
        last=insert_rightpos(item,head);
        break;
case 7:printf("enter key whose duplicates to be deleted\n");
       scanf("%d",&item);
       last=delete_dup_key(item,head);
       break;
case 8:printf("enter key which has to be deleted\n");
       scanf("%d",&item);
       last=delete_value(item,head);
       break;
case 9:printf("enter item\n");
```

```
                scanf("%d",&item);
                search(item,head);
                break;
            case 10: display(head);
                    break;
            case 11:break;
            default:printf("invalid choice");
                 break;
            }
        }while(choice!=11);
return 0;
}
```

Output-



```
enter the item at rear end
31
enter the choice
2
enter the item at rear end
41
enter the choice
10
contents of list
13
12
11
21
31
41

enter the choice
3
the node deleted is 13
enter the choice
4
the node deleted is 41
enter the choice
10
contents of list
12
11
21
31

enter the choice
```

```
enter the choice
5
enter the key item
11
enter towards left of 11=6
enter the choice
6
enter the key item
21
enter towards right of 21=7
enter the choice
10
contents of list
12
6
11
21
7
31

enter the choice
1
enter the item at front end
6
enter the choice
1
enter the item at front end
6
enter the choice
2
enter the item at rear end
```

```
enter the item at rear end
6
enter the choice
10
contents of list
6
6
12
6
11
21
7
31
6

enter the choice
7
enter key whose duplicates to be deleted
6
Duplicates are deleted
enter the choice
10
contents of list
6
12
11
21
7
31

enter the choice
```

```
enter the choice
1
enter the item at front end
11
enter the choice
1
enter the item at front end
11
enter the choice
2
enter the item at rear end
11
enter the choice
10
contents of list
11
11
6
12
11
21
7
31
11

enter the choice
8
enter key which has to be deleted
7
key found at 1 positions and are deleted
enter the choice
```

```
enter the choice
8
enter key which has to be deleted
11
key found at 4 positions and are deleted
enter the choice
10
contents of list
6
12
21
31

enter the choice
9
enter item
12
search successful
enter the choice
9
enter item
1
search unsuccessfull
enter the choice
3
the node deleted is 6
enter the choice
3
the node deleted is 12
enter the choice
4
```

```
12
search successful
enter the choice
9
enter item
1
search unsuccessfull
enter the choice
3
the node deleted is 6
enter the choice
3
the node deleted is 12
enter the choice
4
the node deleted is 31
enter the choice
4
the node deleted is 21
enter the choice
4
list empty
enter the choice
10
list empty
enter the choice
11


...Program finished with exit code 0
Press ENTER to exit console.
```