1BM19CS079

LIKITHA.B

9-12-2020

PROGRAM-1

WAP TO IMPLEMENT STACK AND QUEUES USING LINKED REPRESENTATION.

CODE-

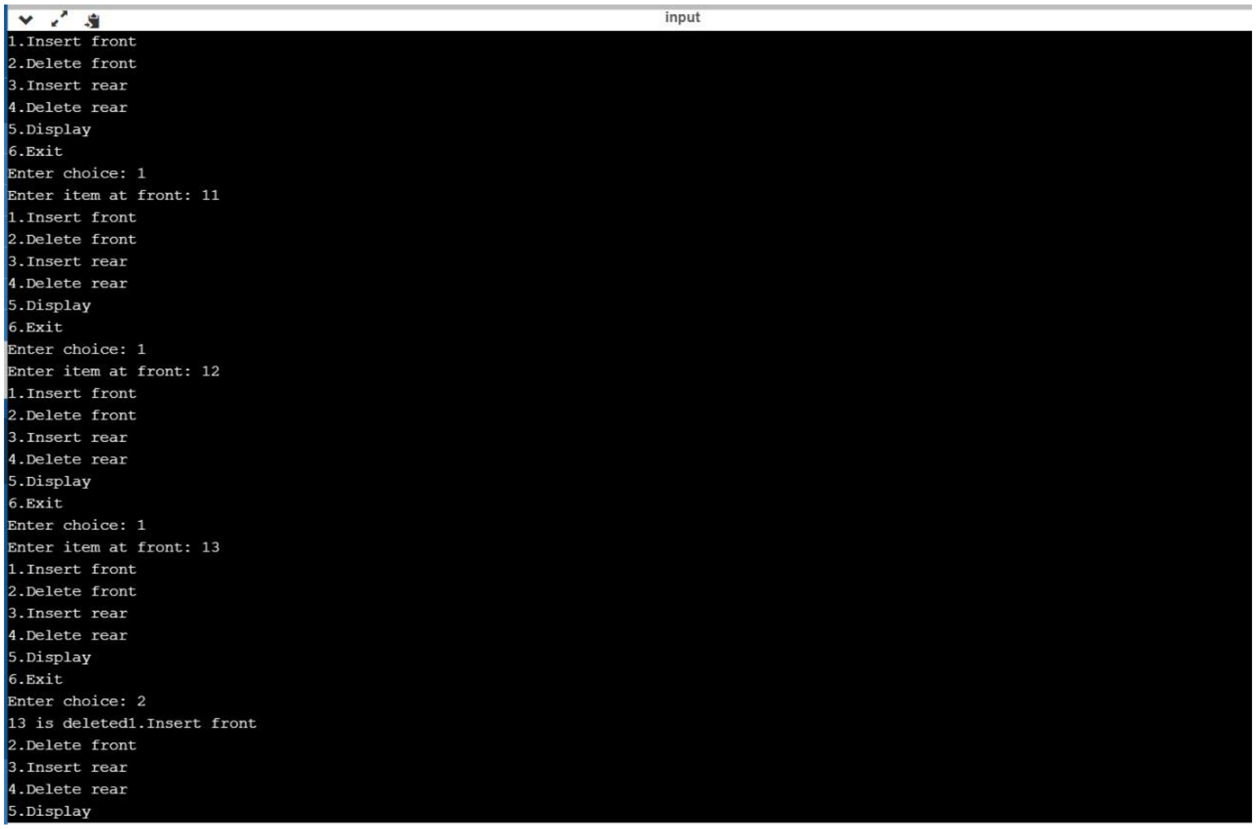
```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

   int data;

   struct node *next;

};

typedef struct node *NODE;

NODE get()

{

   NODE x;

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

   if(x==NULL)

   {

      printf("Memory full\n");

      exit(0);

   }
```

```
    return x;
}
void f(NODE x)
{
    free(x);
}
NODE insert_front(NODE first, int item)
{
    NODE temp;
    temp=get();
    temp->data=item;
    temp->next=NULL;
    if(first==NULL)
    {
        return temp;
    }
    temp->next=first;
    first=temp;
    return first;
}
NODE delete_front(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("List Empty\n");
        return first;
    }
    temp=first;
    temp=temp->next;
    printf("%d is deleted",first->data);
    f(first);
    return temp;
}
NODE insert_rear(NODE first, int item)
{
    NODE temp,cur;
    temp=get();
    temp->data=item;
    temp->next=NULL;
    if(first==NULL)
    {
        return temp;
    }
    cur=first;
    while(cur->next!=NULL)
    {
        cur=cur->next;
        cur->next=temp;
    }
    return first;
}
NODE delete_rear(NODE first)
{
    NODE cur,prev;
    if(first==NULL)
    {
        printf("List Empty\n");
        return first;
    }
    if(first->next==NULL)
    {
        printf("%d is deleted\n",first->data);
        f(first);
        return NULL;
    }
    prev=NULL;
    cur=first;
    while(cur->next!=NULL)
    {
        prev=cur;
        cur=cur->next;
    }
    printf("Item deleted=%d\n",cur->data);
    f(cur);
    prev->next=NULL;
    return first;
}
void display(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("List Empty\n");
    }
    for(temp=first;temp!=NULL;temp=temp->next)
    {
        printf("%d\n",temp->data);
    }
}
int main()
{
    int item,ch;
    NODE first=NULL;
    do{
        printf("1.Insert front\n2.Delete front\n3.Insert rear\n4.Delete rear\n5.Display\n6.Exit\n");
        printf("Enter choice: ");
        scanf("%d",&ch);
        switch(ch)
        {
```

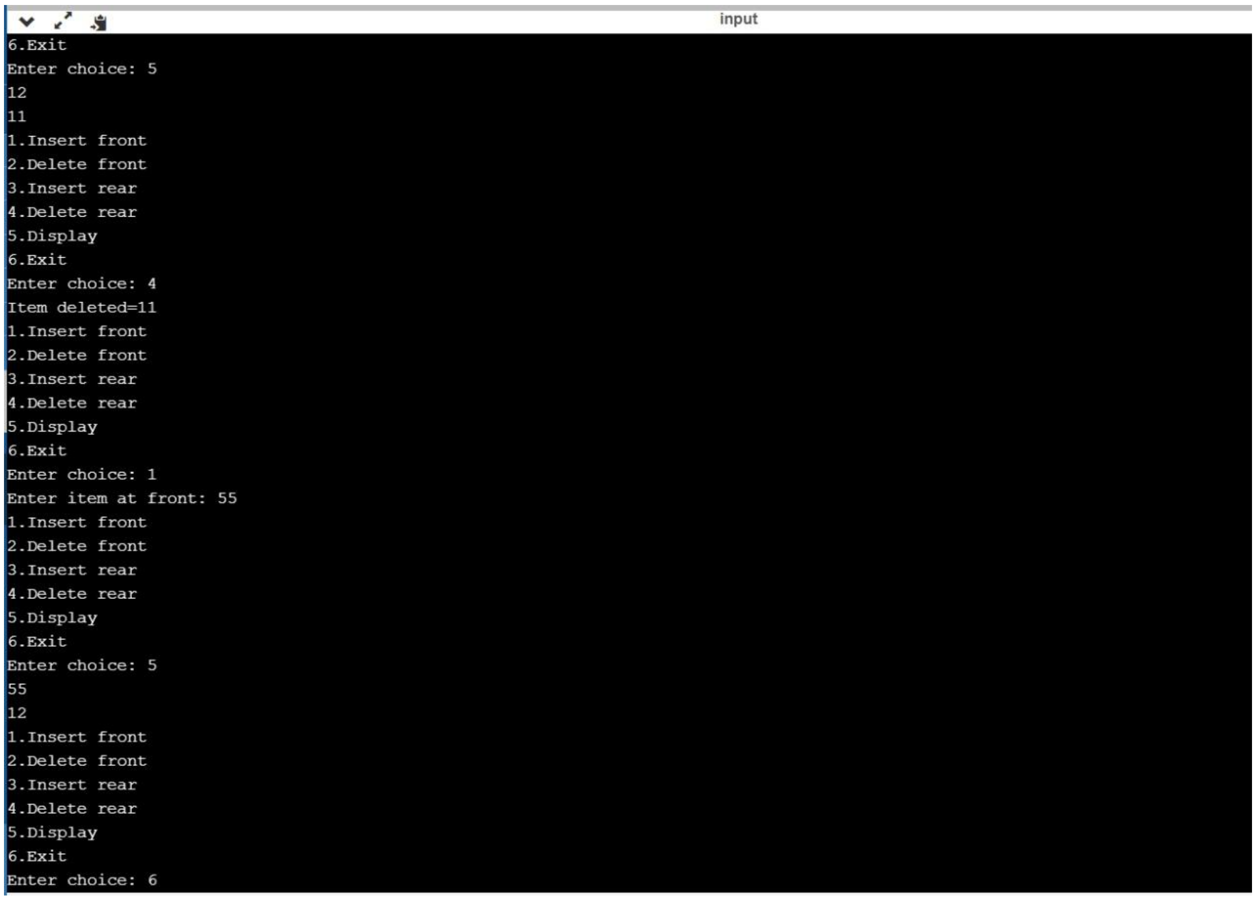
```
  case 1:

  printf("Enter item at front: ");

  scanf("%d",&item);

  first=insert_front(first,item);

  break;

  case 2:

  first=delete_front(first);

  break;

  case 3:

  printf("Enter item at rear: ");

  scanf("%d",&item);

  first=insert_rear(first,item);

  break;

  case 4:

  first=delete_rear(first);

  break;

  case 5:

  display(first);

  break;

  case 6:

  break;

  default:

  printf("\nEnter a valid choice\n");

  break;

}
```

```
    }while(ch!=6);

    return 0;

}
```

OUTPUT-



```
1.Insert front
2.Delete front
3.Insert rear
4.Delete rear
5.Display
6.Exit
Enter choice: 1
Enter item at front: 11
1.Insert front
2.Delete front
3.Insert rear
4.Delete rear
5.Display
6.Exit
Enter choice: 1
Enter item at front: 12
1.Insert front
2.Delete front
3.Insert rear
4.Delete rear
5.Display
6.Exit
Enter choice: 1
Enter item at front: 13
1.Insert front
2.Delete front
3.Insert rear
4.Delete rear
5.Display
6.Exit
Enter choice: 2
13 is deleted1.Insert front
2.Delete front
3.Insert rear
4.Delete rear
5.Display
```



```
6.Exit
Enter choice: 5
12
11
1.Insert front
2.Delete front
3.Insert rear
4.Delete rear
5.Display
6.Exit
Enter choice: 4
Item deleted=11
1.Insert front
2.Delete front
3.Insert rear
4.Delete rear
5.Display
6.Exit
Enter choice: 1
Enter item at front: 55
1.Insert front
2.Delete front
3.Insert rear
4.Delete rear
5.Display
6.Exit
Enter choice: 5
55
12
1.Insert front
2.Delete front
3.Insert rear
4.Delete rear
5.Display
6.Exit
Enter choice: 6
```

PROGRAM-2

DOUBLY LINKED LIST

CODE-

```
#include<stdio.h>

#include<stdlib.h>

#include<process.h>

struct node

 {

  int info;

  struct node *rlink;
  struct node *llink;
 };
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 insert_rear(NODE head,int item)
{
NODE temp,cur;
temp=getnode();
temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
```

```
cur=head->llink;
temp->llink=cur;
cur->rlink=temp;
head->llink=temp;
temp->rlink=head;
head->info=head->info+1;
return head;
}
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_righttpos(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->rlink;
 printf("enter towards left of %d=",item);
 temp=getnode();
 scanf("%d",&temp->info);
 prev->llink=temp;
 temp->llink=cur;
 cur->rlink=temp;
 temp->rlink=prev;
 return head;
}
NODE delete_all_key(int item,NODE head)
{
NODE prev,cur,next;
int count;
   if(head->rlink==head)
    {
     printf("LE");
     return head;
     }
```

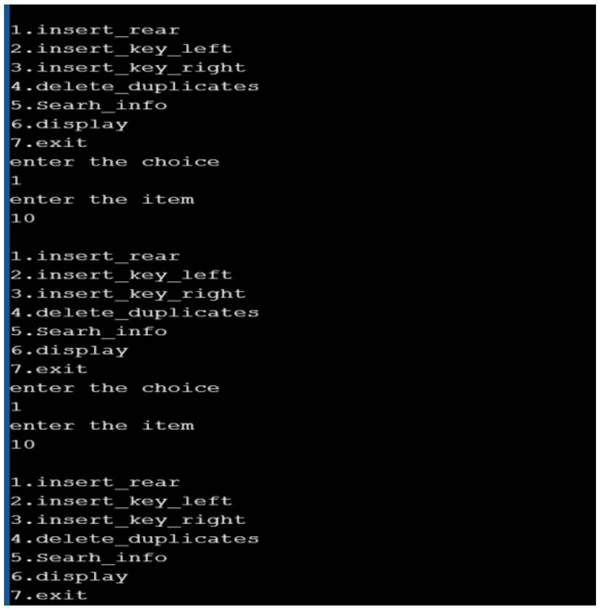
```
count=0;

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);

  cur=next;

 }

}

if(count==0)

  printf("key not found");

else

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


return head;

}

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

```
{
NODE cur;
if(head->rlink==head)
{
printf("list empty\n");
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)
{
   printf("Search Successfull\n");
   break;
}
cur=cur->rlink;
}
if(cur==head)
{
 printf("Info not found\n");
 }
}
void display(NODE head)
{
NODE temp;
if(head->rlink==head)
{
```

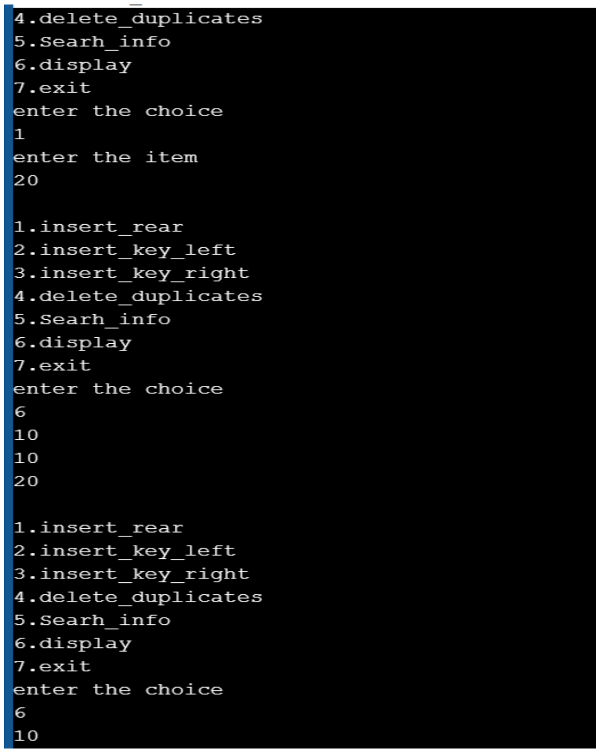
```
printf("list empty\n");

return;

}

for(temp=head->rlink;temp!=head;temp=temp->rlink)

printf("%d\n",temp->info);

}

void main()

{

int item,choice,key;

NODE head;

head=getnode();

head->rlink=head;

head->llink=head;

for(;;)

{

printf("\n1.insert_rear\n2.insert_key_left\n3.insert_key_right\n4.delete_duplicates\n5.Searh_info\n6.d
isplay\n7.exit\n");

printf("enter the choice\n");

scanf("%d",&choice);

switch(choice)

 {

  case 1:printf("enter the item\n");

                  scanf("%d",&item);

                  head=insert_rear(head,item);

                  break;

  case 2:printf("enter the key item\n");
```

```
                scanf("%d",&item);
                head=insert_leftpos(item,head);
                break;
 case 3:printf("enter the key item\n");
                scanf("%d",&item);
                head=insert_righttpos(item,head);
                break;
 case 4:printf("enter the key item\n");
                scanf("%d",&item);
                head=delete_all_key(item,head);
                break;
 case 5:printf("enter the key item\n");
                scanf("%d",&item);
                Search_info(item,head);
                break;
 case 6:display(head);
                break;
 default:exit(0);
                 break;
 }
}
}
```

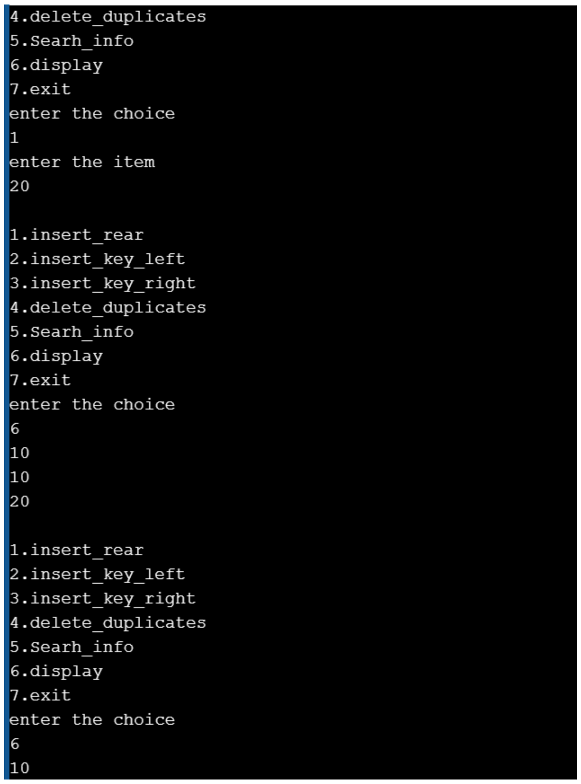
OUTPUT-



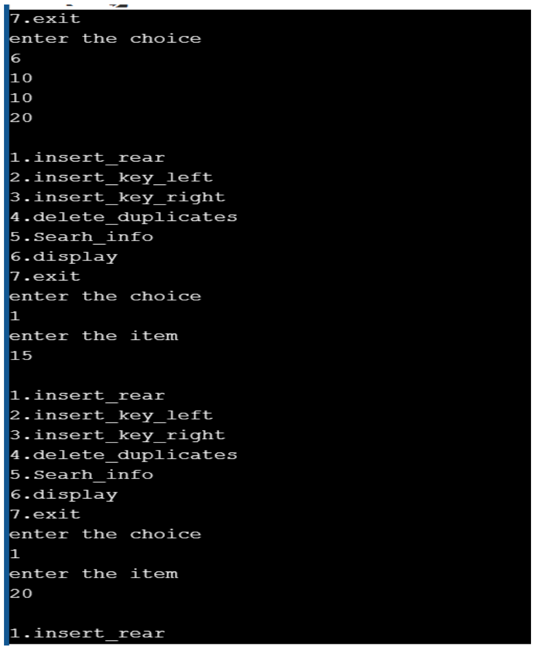
```
1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
10

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
10

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
```



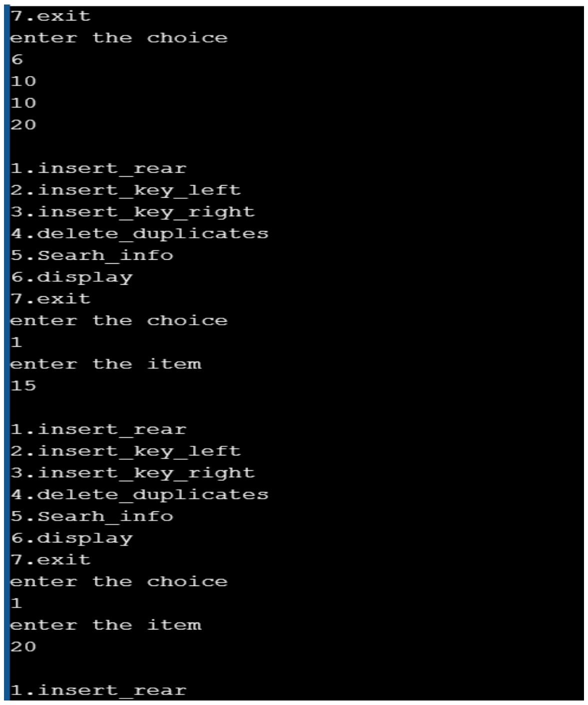
```
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
20

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
6
10
10
20

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
6
10
```



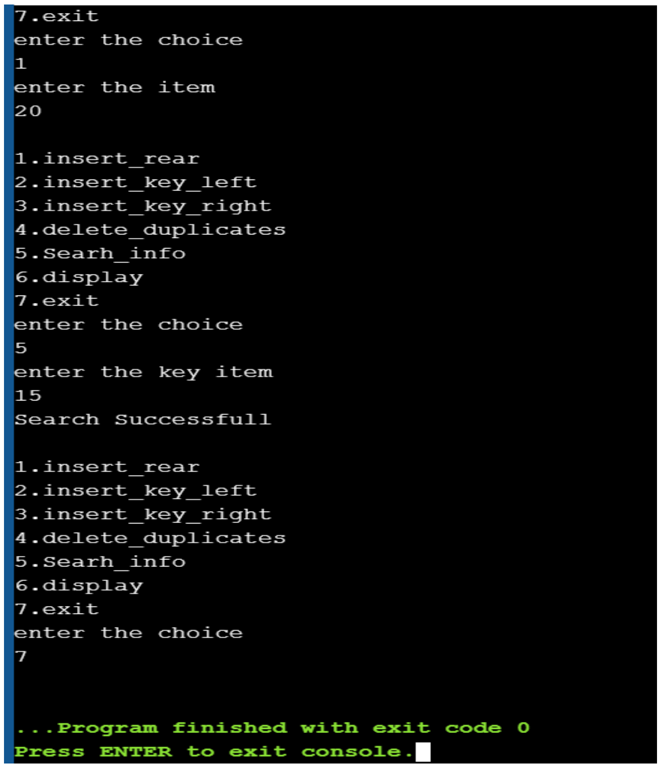
```
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
20

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
6
10
10
20

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
6
10
```



```
7.exit
enter the choice
6
10
10
20

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
15

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
20

1.insert_rear
```



```
7.exit
enter the choice
6
10
10
20

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
15

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
20

1.insert_rear
```



```
7.exit
enter the choice
1
enter the item
20

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
5
enter the key item
15
Search Successfull

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
7


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