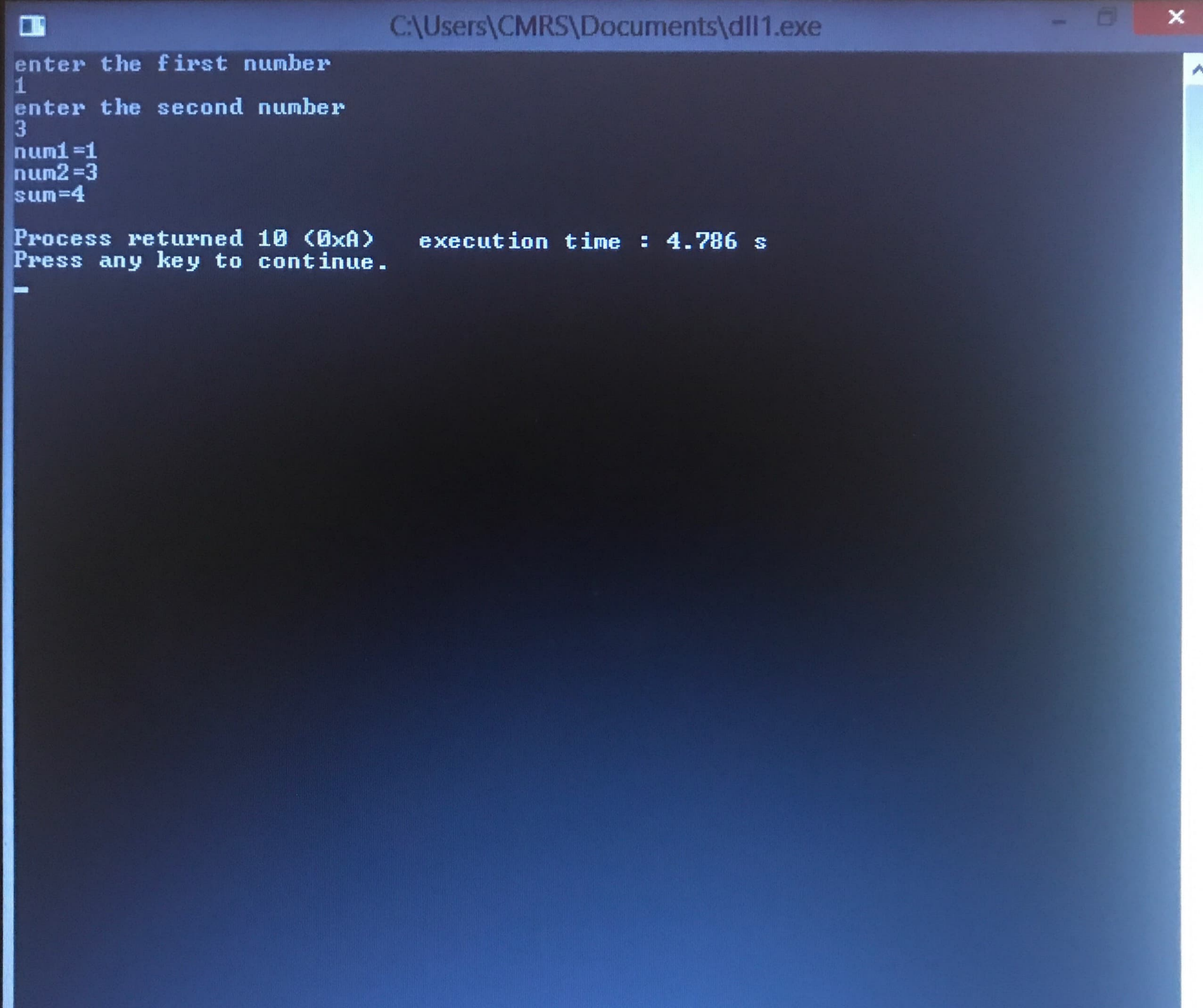
```c
#include<stdio.h>
#include<conio.h>
#include<process.h>
struct node
{
  int info;
  struct node *link;
};
typedef struct node *NODE;
NODE insert_front(int, NODE);
NODE insert_rear(int, NODE );
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
  printf("mem full\n");
  exit(0);
 }
 return x;
}
void freenode(NODE x)
{
free(x);
}
NODE read_number(NODE head)
{
char c;
while((c=getche())!='\r')
{
 if(isdigit(c))
 head=insert_front(c-'0',head);
 else
 {
 printf("invalid integer\n");
```

```
37      exit(1);
38      }
39    }
40    printf("\n");
41    return head;
42    }
43    NODE add_long(NODE h1,NODE h2,NODE h3)
44    {
45      NODE c,c1,c2,h;
46      int sum,carry,digit;
47      carry=0;
48      c1=h1->link;
49      c2=h2->link;
50      while(c1!=h1 && c2!=h2)
51      {
52      sum=c1->info+c2->info+carry;
53      digit=sum%10;
54      carry=sum/10;
55      h3=insert_rear(digit,h3);
56      c1=c1->link;
57      c2=c2->link;
58      }
59      if(c1!=h1)
60      c=c1,h=h1;
61      else
62      c=c2,h=h2;
63      while(c!=h)
64      {
65      sum=c->info+carry;
66      digit=sum%10;
67      carry=sum/10;
68      h3=insert_rear(digit,h3);
69      c=c->link;
70      }
71      if(carry==1)
72      h3=insert_rear(carry,h3);
```

```
 h3=insert_rear(carry,h3);
 return h3;
}
NODE insert_front(int item, NODE last)
{
NODE temp;
temp=getnode();
temp->info=item;
if(last==NULL)
last=temp;
else
temp->link=last->link;
last->link=temp;
return last;
}
NODE insert_rear(int item, NODE last)
{
NODE temp;
temp=getnode();
temp->info=item;
if(last==NULL)
last=temp;
else
temp->link=last->link;
last->link=temp;
return last;
}
void display_number(NODE head)
{
 int k,n,*a;
 NODE cur;
 if(head->link==head)
 {
 printf("number doesnot exist\n");
 return;
 }
```

```
107     }
108     n=head->info;
109     a=(int *)malloc(n*sizeof(int));
110     cur=head->link,k=0;
111     while(cur!=head)
112     {
113     a[k++]=cur->info;
114     cur=cur->link;
115     }
116     while(--k!=-1)
117     printf("%d",a[k]);
118     printf("\n");
119   }
120   void main()
121   {
122   NODE h1,h2,h3;
123   clrscr();
124   h1=getnode();
125   h2=getnode();
126   h3=getnode();
127   h1->link=h1;
128   h2->link=h2;
129   h3->link=h3;
130   h1->info=h2->info=h3->info=0;
131   printf("enter the first number\n");
132   h1=read_number(h1);
133   printf("enter the second number\n");
134   h2=read_number(h2);
135   h3=add_long(h1,h2,h3);
136   printf("num1=");
137   display_number(h1);
138   printf("num2=");
139   display_number(h2);
140   printf("sum=");
141   display_number(h3);
142   }
```

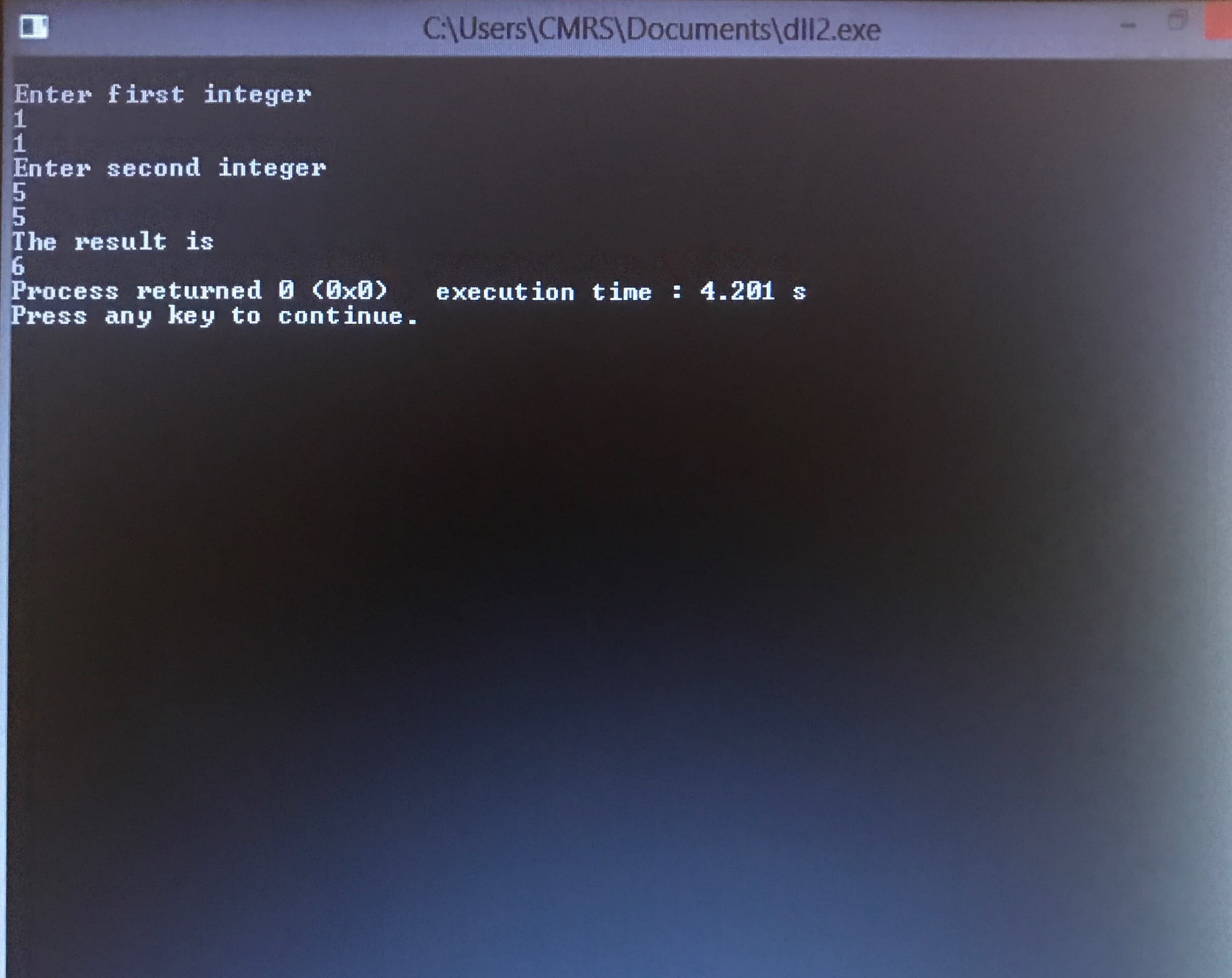

C:\Users\CMRS\Documents\dll1.exe
enter the first number
1
enter the second number
3
num1=1
num2=3
sum=4
Process returned 10 (0xA)   execution time : 4.786 s
Press any key to continue.

```c
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
struct NODE
{
int info;
struct NODE*link;
};
typedef struct NODE*node;
node getnode()
{
node x;
x=(node)malloc(sizeof(struct NODE));
if(x==NULL)
{
printf("out of mamory\n");
exit(0);
}
return x;
}
node ins_front(node first,int item)
{
node temp;
temp=getnode();
temp->info=item;
temp->link=first;
return temp;
}
node extract(char *s,node head)
{
    int i,n;
    for(i=0;i<strlen(s);i++)
    {
    n=s[i]-'0';
    head=ins_front(head,n);
```

```
37      }
38      return head;
39  }
40
41  node addlong(node head1,node head2,node head3)
42  {
43      int temp,sum,carry=0;
44      node cur1,cur2;
45      cur1=head1;
46      cur2=head2;
47      while(cur1!=NULL&&cur2!=NULL)
48      {
49          temp=cur1->info+cur2->info+carry;
50          if(temp>9)
51          {
52          sum=temp%10;
53          carry=temp/10;
54          }
55          else
56          {
57          sum=temp;
58          carry=0;
59          }
60          head3=ins_front(head3,sum);
61          cur1=cur1->link;
62          cur2=cur2->link;
63      }
64      while(cur1!=NULL)
65      {
66      temp=cur1->info+carry;
67      if(temp>9)
68      {
69      sum=temp%10;
70      carry=temp/10;
71      }
72      else
```

```
73      {
74      sum=temp;
75      carry=0;
76      }
77      head3=ins_front(head3,sum);
78      cur1=cur1->link;
79      }
80    while(cur2!=NULL)
81      {
82      temp=cur2->info+carry;
83      if(temp>9)
84      {
85      sum=temp%10;
86      carry=temp/10;
87      }
88      else
89      {
90      sum=temp;
91      carry=0;
92      }
93      head3=ins_front(head3,sum);
94      cur2=cur2->link;
95      }
96
97      if(cur1==NULL&&cur2==NULL)
98      {
99          if(carry==1)
100         head3=ins_front(head3,carry);
101     }
102     return head3;
103   }
104
105
106 void display(node first)
107 {
108 node cur;
```

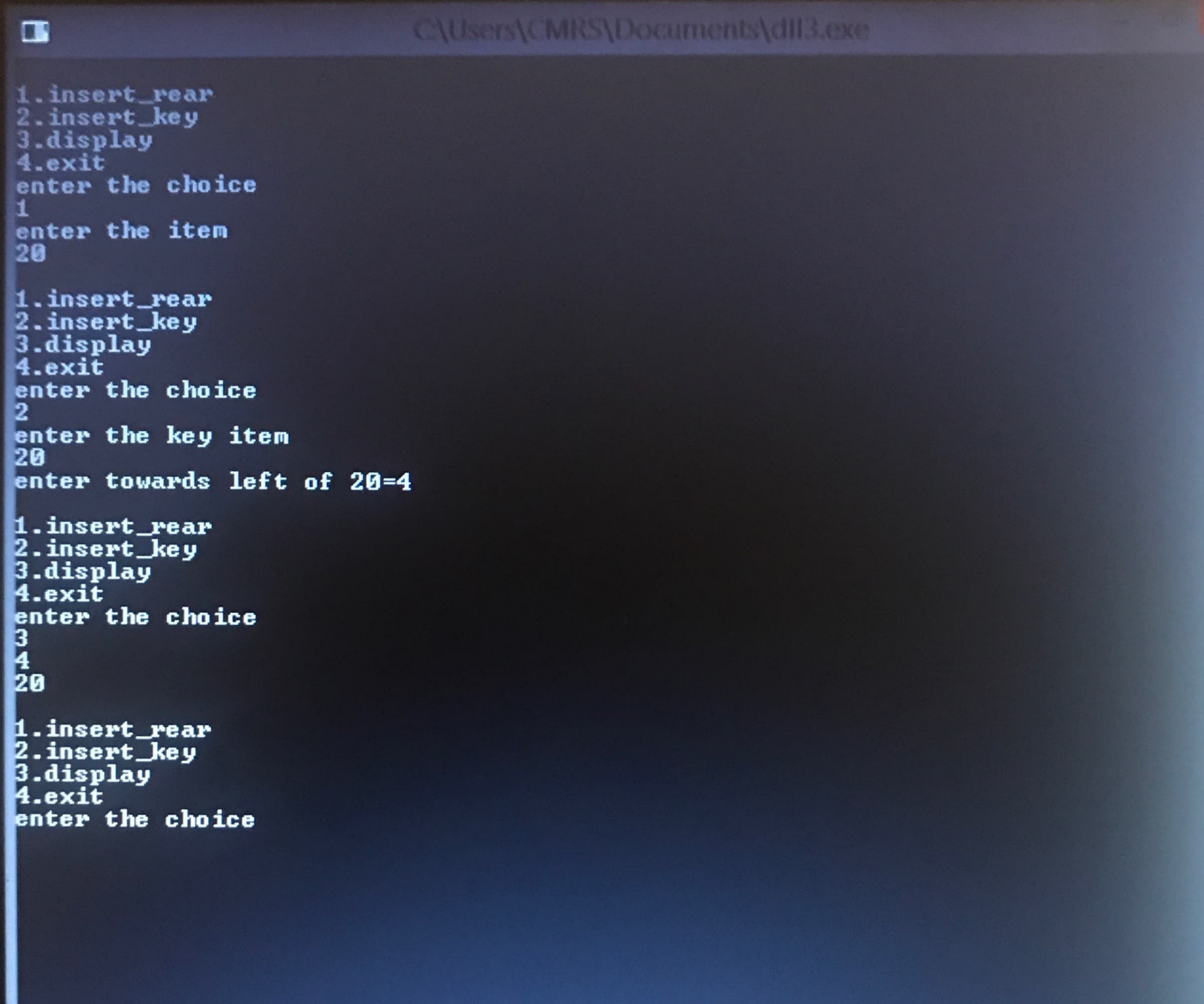
```
{
node cur;
if(first==NULL)
{
printf("Empty\n");
return;
}
cur=first;
while(cur!=NULL)
{
printf("%d\t",cur->info);
cur=cur->link;
}
}
void main()
{
    int ch;
    node head1=NULL;
    node head2=NULL;
    node head3=NULL;
    char s1[30],s2[30];

    printf("\nEnter first integer\n");
    scanf("%s",s1);
    head1=extract(s1,head1);
    display(head1);
    printf("\nEnter second integer\n");
    scanf("%s",s2);
    head2=extract(s2,head2);
    display(head2);
    head3=addlong(head1,head2,head3);
    printf("\nThe result is\n");
    display(head3);
}
```


C:\Users\CMRS\Documents\dll2.exe
Enter first integer
1
1
Enter second integer
5
5
The result is
6
Process returned 0 (0x0)   execution time : 4.201 s
Press any key to continue.

```c
#include<stdio.h>
#include<conio.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;
```

```
37    temp->rlink=head;
38    head->info=head->info+1;
39    return head;
40    }
41    NODE insert_leftpos(int item,NODE head)
42    {
43    NODE temp,cur,prev;
44    if(head->rlink==head)
45    {
46    printf("list empty\n");
47    return head;
48    }
49    cur=head->rlink;
50    while(cur!=head)
51    {
52    if(item==cur->info)break;
53    cur=cur->rlink;
54    }
55    if(cur==head)
56    {
57     printf("key not found\n");
58     return head;
59     }
60     prev=cur->llink;
61     printf("enter towards left of %d=",item);
62     temp=getnode();
63     scanf("%d",&temp->info);
64     prev->rlink=temp;
65     temp->llink=prev;
66     cur->llink=temp;
67     temp->rlink=cur;
68     return head;
69    }
70    void display(NODE head)
71    {
72    NODE temp;
```

```
{
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\n3.display\n4.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:display(head);
         break;
  default:exit(0);
          break;
  }
 }
}
```


C:\Users\CMRS\Documents\dll3.exe
1.insert_rear
2.insert_key
3.display
4.exit
enter the choice
1
enter the item
20
1.insert_rear
2.insert_key
3.display
4.exit
enter the choice
2
enter the key item
20
enter towards left of 20=4
1.insert_rear
2.insert_key
3.display
4.exit
enter the choice
3
4
20
1.insert_rear
2.insert_key
3.display
4.exit
enter the choice

```c
#include<stdio.h>
#include<conio.h>
#include<process.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;
```

```
37      return head;
38      }
39      NODE dinsert_rear(int item,NODE head)
40      {
41      NODE temp,cur;
42      temp=getnode();
43      temp->info=item;
44      cur=head->llink;
45      head->llink=temp;
46      temp->rlink=head;
47      temp->llink=cur;
48      cur->rlink=temp;
49      return head;
50      }
51      NODE ddelete_front(NODE head)
52      {
53      NODE cur,next;
54      if(head->rlink==head)
55      {
56      printf("dq empty\n");
57      return head;
58      }
59      cur=head->rlink;
60      next=cur->rlink;
61      head->rlink=next;
62      next->llink=head;
63      printf("the node deleted is %d",cur->info);
64      freenode(cur);
65      return head;
66      }
67      NODE ddelete_rear(NODE head)
68      {
69      NODE cur,prev;
70      if(head->rlink==head)
71      {
72      printf("dq empty\n");
```

```
73      return head;
74      }
75      cur=head->llink;
76      prev=cur->llink;
77      head->llink=prev;
78      prev->rlink=head;
79      printf("the node deleted is %d",cur->info);
80      freenode(cur);
81      return head;
82      }
83      void display(NODE head)
84      {
85      NODE temp;
86      if(head->rlink==head)
87      {
88      printf("dq empty\n");
89      return;
90      }
91      printf("contents of dq\n");
92      temp=head->rlink;
93      while(temp!=head)
94      {
95      printf("%d",temp->info);
96      temp=temp->rlink;
97      }
98      printf("\n");
99      }
100     void main()
101     {
102     NODE head,last;
103     int item, choice;
104     head=getnode();
105     head->rlink=head;
106     head->llink=head;
107     for(;;)
108     {
```

```
98      printf("\n");
99      }
100     void main()
101     {
102     NODE head,last;
103     int item, choice;
104     head=getnode();
105     head->rlink=head;
106     head->llink=head;
107     for(;;)
108     {
109         printf("\n1:insert front\n2:insert rear\n3:delete front\n4:delete rear\n5:display\n6:exit\n");
110         printf("enter the choice\n");
111         scanf("%d",&choice);
112         switch(choice)
113         {
114             case 1: printf("enter the item at front end\n");
115                 scanf("%d",&item);
116                 last=dinsert_front(item,head);
117                 break;
118             case 2: printf("enter the item at rear end\n");
119                 scanf("%d",&item);
120                 last=dinsert_rear(item,head);
121                 break;
122             case 3:last=ddelete_front(head);
123                 break;
124             case 4: last=ddelete_rear(head);
125                 break;
126             case 5: display(head);
127                 break;
128             default:exit(0);
129             }
130     }
131     }
132
```

```
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:exit
enter the choice
1
enter the item at front end
20

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:exit
enter the choice
2
enter the item at rear end
70

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:exit
enter the choice
1
enter the item at front end
50

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:exit
enter the choice
3
the node deleted is 50
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:exit
enter the choice
5
contents of dq
2070
```

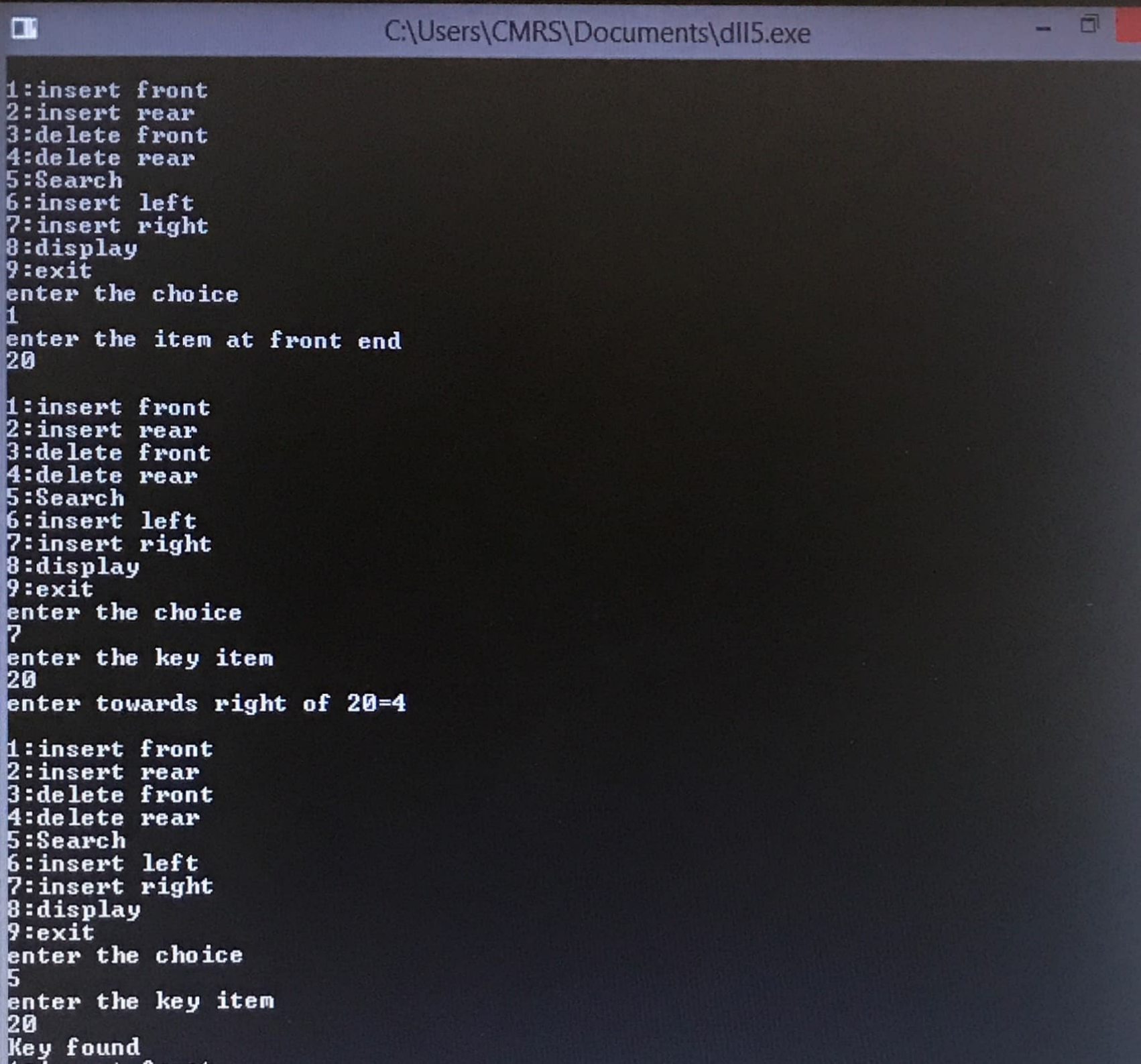
```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
     int info;
     struct node *llink;
     struct node *rlink;
     };
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
        printf("mem full\n");
        exit(0);
        }
    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;
}
```

```
40      temp=getnode();
41      temp->info=item;
42      cur=head->llink;
43      head->llink=temp;
44      temp->rlink=head;
45      temp->llink=cur;
46      cur->rlink=temp;
47      return head;
48      }
49
50  NODE dsearch(int item,NODE head){
51      NODE temp,cur,prev;
52      if(head->rlink==head)
53      {
54      printf("list empty\n");
55      return head;
56      }
57      cur=head->rlink;
58      while(cur!=head)
59      {
60      if(item==cur->info){
61          printf("Key found");
62        break;
63      }
64      cur=cur->rlink;
65      }
66
67      if(cur==head)
68      {
69       printf("key not found\n");
70       return head;
71       }
72      }
73
74      NODE ddelete_front(NODE head)
75      {
```

```
80      return head;
81      }
82      cur=head->rlink;
83      next=cur->rlink;
84      head->rlink=next;
85      next->llink=head;
86      printf("the node deleted is %d",cur->info);
87      freenode(cur);
88      return head;
89      }
90      NODE ddelete_rear(NODE head)
91      {
92      NODE cur,prev;
93      if(head->rlink==head)
94      {
95      printf("dq empty\n");
96      return head;
97      }
98      cur=head->llink;
99      prev=cur->llink;
100     head->llink=prev;
101     prev->rlink=head;
102     printf("the node deleted is %d",cur->info);
103     freenode(cur);
104     return head;
105     }
106     void display(NODE head)
107     {
108     NODE temp;
109     if(head->rlink==head)
110     {
111     printf("dq empty\n");
112     return;
113     }
114     printf("contents of dq\n");
115     temp=head->rlink;
```

```
131     }
132     cur=head->rlink;
133     while(cur!=head)
134     {
135     if(item==cur->info)break;
136     cur=cur->rlink;
137     }
138     if(cur==head)
139     {
140      printf("key not found\n");
141      return head;
142      }
143      prev=cur->llink;
144      printf("enter towards left of %d=",item);
145      temp=getnode();
146      scanf("%d",&temp->info);
147      prev->rlink=temp;
148      temp->llink=prev;
149      cur->llink=temp;
150      temp->rlink=cur;
151      return head;
152     }
153     NODE insert_rightpos(int item,NODE head)
154     {
155     NODE temp,cur,next;
156     if(head->rlink==head)
157     {
158     printf("list empty\n");
159     return head;
160     }
161     cur=head->rlink;
162     while(cur!=head)
163     {
164     if(item==cur->info)break;
165     cur=cur->rlink;
166     }
```

```
{
    printf("\n1:insert front\n2:insert rear\n3:delete front\n4:delete rear\n5:Search\n6:insert left\n7:insert right\n8:display\n9:exit\n");
    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);
         head=dsearch(item,head);
         break;
     case 6:printf("enter the key item\n");
         scanf("%d",&item);
         head=insert_leftpos(item,head);
         break;
    case 7:printf("enter the key item\n");
         scanf("%d",&item);
         head=insert_rightpos(item,head);
         break;
        case 8: display(head);
            break;
        default:exit(0);
        }
    }
}
```


C:\Users\CMRS\Documents\dll5.exe
1:insert front
2:insert rear
3:delete front
4:delete rear
5:Search
6:insert left
7:insert right
8:display
9:exit
enter the choice
1
enter the item at front end
20
1:insert front
2:insert rear
3:delete front
4:delete rear
5:Search
6:insert left
7:insert right
8:display
9:exit
enter the choice
7
enter the key item
20
enter towards right of 20=4
1:insert front
2:insert rear
3:delete front
4:delete rear
5:Search
6:insert left
7:insert right
8:display
9:exit
enter the choice
5
enter the key item
20
Key found