```c
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
struct node{
 float cf;
 float px;
 float py;
 int flag;
 struct node *link;
};
typedef struct node *node;

node getnode()
{
 node x;
 x=(node)malloc(sizeof(struct node));
 if(x == NULL)
 {
   printf("out of memory\n");
   exit(0);
 }
 return x;
}

node insert_rear(float cf,float x,float y,node head)
{
 node temp,cur;
 int flag;
 temp=getnode();
 temp->cf=cf;
 temp->px=x;
 temp->py=y;
 temp->flag=0;
 cur=head->link;
 while(cur->link!=head)
 cur=cur->link;
```

```
38      temp->link=head;
39      return head;
40    }
41
42    node read_poly(node head)
43    {
44      int i;
45      float cf,px,py;
46      printf("Enter the coefficient as -999 to end the polynomial\n");
47      for (i=0;;i++)
48      {
49        printf("enter the %d term\n",i);
50        printf("Coeff:");
51        scanf("%f",&cf);
52        if(cf == -999)
53        break;
54        printf("pow x:");
55        scanf("%f",&px);
56        printf("pow y:");
57        scanf("%f",&py);
58        head=insert_rear(cf,px,py,head);
59      }
60      return head;
61    }
62
63    void display(node head)
64    {
65      node temp;
66      if(head->link==head)
67      {
68        printf("polynomial does not exist\n");
69        return;
70      }
71      temp=head->link;
72      while(temp!=head)
73      {
```

```
73      {
74        printf("%5.2fx^%3.1fy^%3.1f",temp->cf,temp->px,temp->py);
75        temp=temp->link;
76      }
77      printf("\n");
78    }
79
80    node add_poly(node h1,node h2,node h3)
81    {
82      node p1,p2;
83      int x1,x2,y1,y2,cf1,cf2,cf;
84      p1=h1->link;
85      while(p1!=h1)
86      {
87        x1=p1->px;
88        y1=p1->py;
89        cf1=p1->cf;
90
91        p2=h2->link;
92        while(p2!=h2)
93        {
94          x2=p2->px;
95          y2=p2->py;
96          cf2=p2->cf;
97          if(x1==x2 && y1==y2)
98          break;
99          p2=p2->link;
100       }
101       if(p2!=h2)
102       {
103         cf=cf1+cf2;
104         p2->flag=1;
105         if(cf!=0)
106         h3=insert_rear(cf,x1,y1,h3);
107       }
108       else
```

```
109         h3=insert_rear(cf1,x1,y1,h3);
110         p1=p1->link;
111       }
112       p2=h2->link;
113       while(p2!=h2)
114       {
115         if(p2->flag==0)
116         {
117           h3=insert_rear(p2->cf,p2->px,p2->py,h3);
118         }
119         p2=p2->link;
120       }
121       return h3;
122     }
123
124     int main()
125     {
126       node h1,h2,h3;
127       h1=getnode();
128       h2=getnode();
129       h3=getnode();
130       h1->link=h1;
131       h2->link=h2;
132       h3->link=h3;
133       printf("Enter the first polynomial:\n");
134       h1=read_poly(h1);
135       printf("Enter the second polynomial:\n");
136       h2=read_poly(h2);
137       h3=add_poly(h1,h2,h3);
138       printf("The first polynomial\n");
139       display(h1);
140       printf("The second polynomial\n");
141       display(h2);
142       printf("The sum of two polynomials\n");
143       display(h3);
144     }
```

```
Enter the first polynomial:
Enter the coefficient as -999 to end the polynomial
enter the 0 term
Coeff:2
pow x:1
pow y:2
enter the 1 term
Coeff:2
pow x:1
pow y:3
enter the 2 term
Coeff:2
pow x:3
pow y:1
enter the 3 term
Coeff:-999
Enter the second polynomial:
Enter the coefficient as -999 to end the polynomial
enter the 0 term
Coeff:1
pow x:2
pow y:3
enter the 1 term
Coeff:4
pow x:2
pow y:1
enter the 2 term
Coeff:2
pow x:1
pow y:1
enter the 3 term
Coeff:-999
The first polynomial
 2.00x^1.0y^2.0 2.00x^1.0y^3.0 2.00x^3.0y^1.0
The second polynomial
 1.00x^2.0y^3.0 4.00x^2.0y^1.0 2.00x^1.0y^1.0
The sum of two polynomials
 2.00x^1.0y^2.0 2.00x^1.0y^3.0 2.00x^3.0y^1.0 1.00x^2.0y^3.0 4.00x^2.0y^1.0 2.00
x^1.0y^1.0

Process returned 0 (0x0)   execution time : 35.476 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(NODE root,int item)
{
NODE temp,cur,prev;
temp=getnode();
temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
if(root==NULL)
 return temp;
prev=NULL;
cur=root;
```

```
37    while(cur!=NULL)
38    {
39    prev=cur;
40    cur=(item<cur->info)?cur->llink:cur->rlink;
41    }
42    if(item<prev->info)
43     prev->llink=temp;
44    else
45     prev->rlink=temp;
46    return root;
47    }
48    void display(NODE root,int i)
49    {
50    int j;
51    if(root!=NULL)
52     {
53      display(root->rlink,i+1);
54      for(j=0;j<i;j++)
55          printf("   ");
56       printf("%d\n",root->info);
57         display(root->llink,i+1);
58     }
59    }
60    NODE delete(NODE root,int item)
61    {
62    NODE cur,parent,q,suc;
63    if(root==NULL)
64    {
65    printf("empty\n");
66    return root;
67    }
68    parent=NULL;
69    cur=root;
70    while(cur!=NULL&&item!=cur->info)
71    {
72    parent=cur;
```

```
73      cur=(item<cur->info)?cur->llink:cur->rlink;
74      }
75      if(cur==NULL)
76      {
77       printf("not found\n");
78       return root;
79      }
80      if(cur->llink==NULL)
81       q=cur->rlink;
82      else if(cur->rlink==NULL)
83       q=cur->llink;
84      else
85       {
86       suc=cur->rlink;
87       while(suc->llink!=NULL)
88        suc=suc->llink;
89       suc->llink=cur->llink;
90       q=cur->rlink;
91       }
92       if(parent==NULL)
93        return q;
94       if(cur==parent->llink)
95        parent->llink=q;
96       else
97        parent->rlink=q;
98       freenode(cur);
99       return root;
100      }
101
102     void preorder(NODE root)
103     {
104     if(root!=NULL)
105      {
106       printf("%d\n",root->info);
107       preorder(root->llink);
108       preorder(root->rlink);
```

```
    preorder(root->rlink);
   }
  }
 void postorder(NODE root)
{
if(root!=NULL)
 {

  postorder(root->llink);
  postorder(root->rlink);
  printf("%d\n",root->info);
   }
  }
 void inorder(NODE root)
{
if(root!=NULL)
 {

  inorder(root->llink);
  printf("%d\n",root->info);
  inorder(root->rlink);
   }
  }
 void main()
{
int item,choice;
NODE root=NULL;
clrscr();
for(;;)
{
printf("\n1.insert\n2.display\n3.pre\n4.post\n5.in\n6.delete\n7.exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("enter the item\n");
```

```
130   }
131   void main()
132   {
133   int item,choice;
134   NODE root=NULL;
135   clrscr();
136   for(;;)
137   {
138   printf("\n1.insert\n2.display\n3.pre\n4.post\n5.in\n6.delete\n7.exit\n");
139   printf("enter the choice\n");
140   scanf("%d",&choice);
141   switch(choice)
142    {
143     case 1:printf("enter the item\n");
144            scanf("%d",&item);
145            root=insert(root,item);
146            break;
147     case 2:display(root,0);
148            break;
149     case 3:preorder(root);
150            break;
151     case 4:postorder(root);
152            break;
153     case 5:inorder(root);
154           break;
155     case 6:printf("enter the item\n");
156           scanf("%d",&item);
157           root=delete(root,item);
158          break;
159     default:exit(0);
160            break;
161        }
162     }
163   }
164
```

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
20

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
10

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
30

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
40

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
```

```
enter the choice
2
    40
  30
20
  10

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
3
20
10
30
40

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
10
40
30
20

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
10
20
30
40

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
```

```
enter the choice
5
10
20
30
40

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
6
enter the item
10

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
    40
  30
20

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
```

```c
#include<stdio.h>
#include<conio.h>
#include<process.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("memory not available");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert(int item,NODE root)
{
NODE temp,cur,prev;
char direction[10];
int i;
temp=getnode();
temp->info=item;
temp->llink=NULL;
temp->rlink=NULL;
if(root==NULL)
 return temp;
```

```
 return temp;
printf("give direction to insert\n");
scanf("%s",direction);
prev=NULL;
cur=root;
for(i=0;i<strlen(direction)&&cur!=NULL;i++)
{
prev=cur;
if(direction[i]=='1')
cur=cur->llink;
else
cur=cur->rlink;
}
if(cur!=NULL||i!=strlen(direction))
{
printf("insertion not possible\n");
freenode(temp);
return(root);
}
if(cur==NULL)
{
if(direction[i-1]=='1')
prev->llink=temp;
else
prev->rlink=temp;
}
return(root);
}
void preorder(NODE root)
{
if(root!=NULL)
{
printf("the item is %d\n",root->info);
preorder(root->llink);
preorder(root->rlink);
}
```

```
 73    void inorder(NODE root)
 74    {
 75    if(root!=NULL)
 76    {
 77    inorder(root->llink);
 78    printf("the item is%d\n",root->info);
 79    inorder(root->rlink);
 80    }
 81    }
 82    void postorder(NODE root)
 83    {
 84    if (root!=NULL)
 85    {
 86    postorder(root->llink);
 87    postorder(root->rlink);
 88    printf("the item is%d\n",root->info);
 89    }
 90    }
 91    void display(NODE root,int i)
 92    {
 93    int j;
 94    if(root!=NULL)
 95    {
 96    display(root->rlink,i+1);
 97    for (j=1;j<=i;j++)
 98    printf("   ");
 99    printf("%d\n",root->info);
100    display(root->llink,i+1);
101    }
102    }
103
104    void main()
105    {
106    NODE root=NULL;
107    int choice,i,item;
108    clrscr();
```

```
for(;;)
{
printf("1.insert\n2.preorder\n3.inorder\n4.postorder\n5.display\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("enter the item\n");
        scanf("%d",&item);
        root=insert(item,root);
        break;
case 2: if(root==NULL)
        {
         printf("tree is empty");
        }
        else
        {
         printf("given tree is");
         display(root,1);
         printf("the preorder traversal is \n");
         preorder(root);
        }
        break;
case 3:if(root==NULL)
       {
        printf("tree is empty");
       }
       else
       {
        printf("given tree is");
        display(root,1);
        printf("the inorder traversal is \n");
        inorder(root);
        }
       break;
case 4:if (root==NULL)
```

```
128             printf("the preorder traversal is \n");
129             preorder(root);
130            }
131            break;
132  case 3:if(root==NULL)
133           {
134            printf("tree is empty");
135           }
136           else
137           {
138            printf("given tree is");
139            display(root,1);
140            printf("the inorder traversal is \n");
141            inorder(root);
142           }
143          break;
144  case 4:if (root==NULL)
145           {
146           printf("tree is empty");
147            }
148          else
149          {
150           printf("given tree is");
151           display(root,1);
152           printf("the postorder traversal is \n");
153           postorder(root);
154          }
155        break;
156  case 5:display(root,1);
157         break;
158  default:exit(0);
159  }
160  }
161  }
162
```

```
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
20
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
30
give direction to insert
1
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
20
give direction to insert
3
insertion not possible
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
60
give direction to insert
5
insertion not possible
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
70
give direction to insert
2
insertion not possible
1.insert
```

```
60
give direction to insert
5
insertion not possible
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
70
give direction to insert
2
insertion not possible
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
2
given tree is    30
  20
the preorder traversal is
the item is 20
the item is 30
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
4
given tree is    30
  20
the postorder traversal is
the item is30
the item is20
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
5
    30
  20
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
```