```
struct node {
 int data;
 struct node right; left; };
 typedef struct node *NODE;
 void treetoList (Node root, node & prew,
                        node & head) {

 if (!root) return o;
  treetoList (root →left, prew, head);
  root →left = prev;
  if (prev) { prev →right = root;
  else head = root;
  Node rignt = root →right;
  head → left = root;
  roo t →right = head;
  pre v = root;
   treetoList (rignt, prew, head) }
 Node treetoList (Node root)

 {Node.prev = NULL;
   Node.head = NULL;
 treetoList (root, prev, heat) ; }
 return head; }
```

Sayilari Agaca yerk. Prog.

```
Struct nodetype {
int info;
struct nodetype *left;
struct nodetype *right; }
typedef struct nodetyp. *NODEPTR;
main()
NODEPTR p,tree;
NODEPTR P,q;
int number;
scanf("%d",&number);
ptree=maketree (number);
while (scanf("%d", &number)! =EOF ){
P=q = ptree;
while (number! = P->info && q! =NULL ){
P=q;
if (number < p->info)
   q =p->left; else
   q = p->right;
if (number == p->info)
{ printf ("%d is a duplicate \n",number);
else if (number (p->info)
   setleft (p,number);
else
   setright(p,number); } }
```

Kuyruktan İkili Ağaca Aktarma

```c
struct kuyruk {
  int info;
  struct kuyruk *next; };
typedef struct kuyruk *KUYRUKPTR;
struct tree {
  int info;
  struct tree *left;
  struct tree *right; };
typedef struct tree *TREENODEPTR;
void kuyruktotree (KUYRUKPTR *bas,
  KUYRUKPTR *son, TREENODEPTR *treePtr)
{ Kuyruk currentPtr;
   int deger;
   currentPtr = *bas;
   while (currentPtr != NULL)
   { deger = remove(bas,son);
     insertTree (treeptr, deger);
     currentPtr = *bas; } }
```

```c
struct node {
int data;
struct node right ;left ; } ;
typedef struct node * NODE ;
void treetoList (Node root, node & prev,
                        node & head ) {
    if (! root ) return 0;
    treetoList (root →left ,prev,head );
    root →left = prev ;
    if (prev) { prev →right =root ;
    else head = root ;
    Node rignt =root →right ;
    head → left = root;
    root →right = head ;
    prev = root;
    treetoList (rignt, prev,head ) }
Node treetoList (Node root)

    { Node .prev = NULL ;
      Node .head = NULL ;
    treetoList (root ,prev, heat ) ;}
    return head ; }
```

Yığın(stack)lon → Tek bağlı. list. okttanma

```c
struct node {
    int info;
    struct node *next; };
typedef struct stack {
    int top;
    int items[10]; } stack;
    typedef struct node *NODEPTR;
    void insert (NODEPTR p, int x)
    { NODEPTR q;
    if (p == NULL() @
    {printf ("gecersiz ettene"); exit (0); }
    q= getnode();
    q →info = x;
    q →next = p → next;
    p →next = q; }
```

```c
typedef struct node {
int data;
struct node *left, *right; } node;
void mirror (node *root) {
if (root == NULL) return;
node *temp;
node * q[100] = {NULL};
int i, j = 0;
while (root)
{ temp = root->left;
root->left = root->right;
root->right = temp;
if (root->left)
q[i++] = root->left;
if (root->right)
q[i++] = root->right;
root = q[j++]; } }
```

Ağaan Tüm Düğümündeki eleman
sayısı toplamı

```c
int main() {
int Tora (NODEPTR tree) {
int toplam=0; if (tree != NULL) {
Tora(tree -)left);
Tora(tree -)right);
toplam++; } }
return toplam }
int main() { NODEPTR tree;
int eleman;
eleman = Tora(tree);
printf("eleman sayısı = "%d" eleman);
return 0; }
```

İkili Ağacın max elemanını Bulan Prog

```c
struct tree {int data; struct tree *right, left; }
*ptree
int maxsayi(ptree *ptree, int max) {
if (ptree == null)
return max;
if (ptree ->data > max)
max = ptree ->data;
maxsayi (ptree->left, max);
maxsayi (ptree->right, max);
return max; }
```

Matrisin satirini Dizide Siralama

```c
int main() {
int i=0, j=0, k=0, temp, min;
int matris [N] [N] = {15,3,23,54,2,53,861,23};
int d[N];
printf ("Mat. boyutu: %d\n kacinosatini siralinak bilyorsak :", N);
scanf ("%d", &k);
for (i=0; i<N; i++)
 d[i] = matris[k-1], [i];
for (i=0; i<N-1; i++) {
min = i;
for (j=i+1; j<N; j++)
 if (d[j] < d[min])
 min = j;
 if (min != i) {
temp = d[min];
 d[min] = d[i];
 d[i] = temp;   } }
printf ("\n\n siralanmis dizi:");
for (i=0; i<N; i++)
printf (" %d", d[i]);
printf (" ");
return 0;   }
```

İkili Ağacın Top. Eleman sayısını Bulan Prog.

```
struct tree {int data', struct tree *right, left;}
*ptree
int toplam;
int toplamsayi(ptree *ptree) {
toplamsayi(ptree ->left);
toplam + +;
toplamsayi(ptree ->right)'; }
```

İkili ağacın sağ ve sol kollarının sayısını
ayrı ayrı bulan prog.

```
struct tree {int data', struct tree *right, left;}
*ptree
int sol', int sag;
int derinlik(ptree *ptree) {
if(ptree = =null)
return 0;
int right = derinlik(ptree ->right);
int left = derinlik(ptree ->left);
sol = left + 1
sag = right + 1'; }
```

Tek bagli Listeyi Tersine Ceviron

```
struct node {int info; struct node *next} *s
void cevir() {
struct node *ptr1, *ptr2, *ptr3;
if (s == NULL)
  printf("bos");
if (s->next == null)
printf("tek elemanli")
ptr1 = s;
ptr2 = ptr1 ->next;
ptr3 = ptr1 ->next;
ptr1 ->next = null;
ptr2 ->next = ptr1;
while (ptr3! = NULL) {
ptr1 = ptr2; ptr2 = ptr3;
ptr3 = ptr3 ->next;
ptr2 ->next = ptr1; }
s = ptr2; }
```

Ağaçtan kuyruğa aktarma

```c
struct tree {
  int info;
  struct tree *left, *right; };
typedef struct tree *TREEPTR;
struct kuyruk { int info;
  struct kuyruk *bas, *son, *next; };
typedef struct kuyruk *KUYRUKPTR;
int agacelemansayisi (TREEPTR *treePtr){
if (*treePtr != NULL)
    return (1 + agacelemansayisi (*treePtr -> left)+
           agacelemansayisi (*treePtr -> right);
    else  return 1; }

int kuyrugaEkle (KUYRUKPTR *bas, *son, int deger){
  KUYRUKPTR newPtr;
  newPtr = (KUYRUKPTR)malloc (sizeof(struct kuyruk));
  newPtr -> info = degeri
  newPtr -> next = NULL;
  if (*bas == NULL)
      *bas = newPtr);
    else (*son -> next) = newPtr;
          *son = newPtr; }                      *treePtr){
int tree ToQueue (KUYRUKPTR *bas "son, *AGACPTR
while (*treePtr != NULL) {
kuyrugaEkle (& *bas, & *son, treePtr -> info)); }}
```

```
int main() {
    AGACPTR *treePtr;
    MuyukPTR *LuyrukPtr;
    int N;
    N = agacelemansayisi(&*treePtr);
    for (int i=0; i<N, i++) {
        tree to Queue(&bos, &son, treePtr->info);
        *bos = *bos->next;
        *treePtr = *treePtr->next;
    }
}
```

İkili Ağaçtan → Tek bağlı Listeye

```c
struct tekbagli {
  int info;
  struct tekbagli *next; };
  typedef struct tekbagli *TEKBAGLIPTR;
  void tree toList (AGACPTR treptr, )
                    TEKBAGLIPTR *listPtr )
{ if (treePtr != NULL)
  { tekbagliyecle (listPtr, treePtr->info);
    tree tolist (treePtr->left, listPtr );
    tree tolist (treePtr->right, listPtr); }}
```