Problem 1

```
#include<iostream>
using namespace std;
typedef   char TElemType;
#define MAXSIZE   10000
typedef struct Node
{
    TElemType  Data;
    Node *lchild;
    Node *rchild;
}*BiTree;
struct SqQueue {
    BiTree *base;
    int front, rear;
};
void IniBiTree(BiTree &T)
{
    T = NULL;
}
void DestroyBiTree(BiTree &T)
{
    if (T)
    {
        if (T->lchild)
            DestroyBiTree(T->lchild);
        if (T->rchild)
            DestroyBiTree(T->rchild);
        delete T;
        T = NULL;
    }
}
void CreatBiTree(BiTree &T)
{
    TElemType c;
    cin >> c;
    if (c == '#')
        T = NULL;
    else
    {
        T = new Node;
        if (!T)
            exit(-1);
        T->Data = c;
        CreatBiTree(T->lchild);
        CreatBiTree(T->rchild);
    }
}
void PreOrderTraverse(BiTree &T)
{
    if (!T)
        return;
    else
    {
        cout << T->Data;
        PreOrderTraverse(T->lchild);
        PreOrderTraverse(T->rchild);
    }
}
void InOrderTraverse(BiTree &T)
{
    if (T)
    {
        InOrderTraverse(T->lchild);
        cout << T->Data;
        InOrderTraverse(T->rchild);
    }
}
void PostOrderTraverse(BiTree &T)
{
    if (T)
    {
        PostOrderTraverse(T->lchild);
        PostOrderTraverse(T->rchild);
        cout << T->Data;
    }
}
void LevelOrderTraverse(BiTree &T)
{
    BiTree p;
    SqQueue Q;
    Q.base = new BiTree[MAXSIZE];
    if (!Q.base) exit(-1);
    Q.front = Q.rear = 0;
    if (T)
    {
        Q.base[Q.rear] = T;
        Q.rear = (Q.rear + 1) % MAXSIZE;
```

```cpp
        while (Q.front != Q.rear)
        {
            p = Q.base[Q.front];
            cout << p->Data;
            Q.front = (Q.front + 1) % MAXSIZE;
            if (p->lchild)
            {
                Q.base[Q.rear] = (p->lchild);
                Q.rear = (Q.rear + 1) % MAXSIZE;
            }
            if (p->rchild)
            {
                Q.base[Q.rear] = (p->rchild);
                Q.rear = (Q.rear + 1) % MAXSIZE;
            }
        }
    }
}
void ShapeBiTree(BiTree &T, int Depth)
{
    if (T)
    {
        ShapeBiTree(T->rchild, Depth + 1);
        for (int i = 1; i < Depth; i++)
            cout << "    ";
        cout << T->Data;
        cout << endl;
        ShapeBiTree(T->lchild, Depth + 1);
    }
}
int main()
{
    BiTree T;
    IniBiTree(T);
    CreatBiTree(T);
    PreOrderTraverse(T);
    cout << endl;
    InOrderTraverse(T);
    cout << endl;
    PostOrderTraverse(T);
    cout << endl;
    LevelOrderTraverse(T);
    cout << endl;
    ShapeBiTree(T, 1);
    DestroyBiTree(T);
    return 0;
}
```

Problem 2

```
#include<iostream>
#include<cmath>
using namespace std;
typedef  char TElemType;
int Tdepth = 1;
int NumLeave = 0;
int NumNode = 0;
typedef struct Node
{
    TElemType  Data;
    Node *lchild;
    Node *rchild;
}*BiTree;
void IniBiTree(BiTree &T)
{
    T = NULL;
}
void DestroyBiTree(BiTree &T)
{
    if (T)
    {
        if (T->lchild)
            DestroyBiTree(T->lchild);
        if (T->rchild)
            DestroyBiTree(T->rchild);
        delete T;
        T = NULL;
    }
}
void CreatBiTree(BiTree &T)
{
    TElemType c;
    cin >> c;
    if (c == '#')
        T = NULL;
    else
    {
        T = new Node;
        if (!T)
            exit(-1);
        T->Data = c;
        CreatBiTree(T->lchild);
        CreatBiTree(T->rchild);
    }
}
void CopyTree(BiTree &T, BiTree &S)
{
    if (T)
    {
        S = new Node;
        if (!S)
            exit(-1);
        S->lchild = S->rchild = NULL;
        S->Data = T->Data;
        CopyTree(T->lchild, S->lchild);
        CopyTree(T->rchild, S->rchild);
    }
}
void Exchange(BiTree &T)
{
    BiTree S;
    if (T)
    {
        S = T->lchild;
        T->lchild = T->rchild;
        T->rchild = S;
        Exchange(T->lchild);
        Exchange(T->rchild);
    }
}
int max(int a, int b)
{
    if (a > b) return a;
    return b;
}
void CalCulBiTree(BiTree &T, int Depth)
{
    if (T)
    {
        NumNode++;
        if (!T->lchild && !T->rchild)
        {
            NumLeave++;
            Tdepth = max(Tdepth, Depth);
        }
        CalCulBiTree(T->lchild, Depth + 1);
        CalCulBiTree(T->rchild, Depth + 1);
    }
}
void ShapeBiTree(BiTree &T, int Depth)
{
    if (T)
    {
        ShapeBiTree(T->rchild, Depth + 1);
        for (int i = 1; i < Depth; i++)
            cout << "     ";
        cout << T->Data;
        cout << endl;
        ShapeBiTree(T->lchild, Depth + 1);
    }
}
int main()
{
    BiTree T, S;
    IniBiTree(T);
    CreatBiTree(T);
    CalCulBiTree(T, 1);
    cout << NumLeave << endl;
    cout << NumNode << endl;
    cout << Tdepth << endl;
    CopyTree(T, S);
    Exchange(S);
    ShapeBiTree(T, 1);
    ShapeBiTree(S, 1);
    DestroyBiTree(T);
    DestroyBiTree(S);
    return 0;
}
```

Problem 3

```
#include<iostream>
using namespace std;
typedef  char TElemType;
typedef int Status;
#define  STACK_INIT_SIZE  100
#define OK     1
#define ERROR  0
#define TRUE   1
#define FALSE  0
typedef struct Node
{
   TElemType  Data;
   Node *lchild;
   Node *rchild;
}*BiTree;
typedef struct {
   BiTree    *base;    //存放动态申请空间的首地址（栈底）
   BiTree   *top;    //栈顶指针
   int    stacksize;     //当前分配的元素个数
} SqStack;
Status InitStack(SqStack &s)
{
   s.base = new BiTree[STACK_INIT_SIZE];
   if (!s.base) exit(-1);
   s.top = s.base;
   s.stacksize = STACK_INIT_SIZE;
   return OK;
}
Status StackEmpty(SqStack s)
{
   if (s.top == s.base)
      return TRUE;
   else
      return FALSE;
}
Status Pop(SqStack &s, BiTree  &e)
{
   if (s.top == s.base)
      return ERROR;
   e = *--s.top;
   return OK;
}
Status Push(SqStack &s, BiTree &e)
{
   *s.top++ = e;
   return OK;
}
```

```
void IniBiTree(BiTree &T)
{
    T = NULL;
}
void DestroyBiTree(BiTree &T)
{
    if (T)
    {
        if (T->lchild)
            DestroyBiTree(T->lchild);
        if (T->rchild)
            DestroyBiTree(T->rchild);
        delete T;
        T = NULL;
    }
}
void CreatBiTree(BiTree &T)
{
    TElemType c;
    cin >> c;
    if (c == '#')
        T = NULL;
    else
    {
        T = new Node;
        if (!T)
            exit(-1);
        T->Data = c;
        CreatBiTree(T->lchild);
        CreatBiTree(T->rchild);
    }
}
void InorderTraverse(BiTree &T)
{
    SqStack S;
    BiTree p = T;
    InitStack(S);
    while (p || !StackEmpty(S))
    {
        if (p)
        {
            Push(S, p);
            cout << "push " << p->Data << endl;
            p = p->lchild;
        }
        else
        {
            Pop(S, p);
            cout << "pop" << endl;
            cout << p->Data << endl;
            p = p->rchild;
        }
    }


}
int main()
{
    BiTree T;
    IniBiTree(T);
    CreatBiTree(T);
    InorderTraverse(T);
    DestroyBiTree(T);
    return 0;
}
```