```
#include <stdio.h>
#include <assert.h>
#include <unistd.h>
#include <stdlib.h>

typedef struct _Tnode {
        char s[11];
        struct _Tnode *r,*l;
} Tnode;

void insert_tree(Tnode **n,char *word,char  *path,int *pathlen)
{
        int res;

        if ((*n)==NULL) {
                *n=(Tnode*)malloc(sizeof(Tnode));
                strcpy((*n)->s,word);
                (*n)->l=(*n)->r=NULL;
                *path=0;
                return;
        }

        res=strcmp((*n)->s,word);

        if (res==0) {
                *pathlen=-1;
                return;
        }

        else if (res<0) {
                *path='R';
                (*pathlen)++;
                insert_tree(&((*n)->r),word,path+1,pathlen);
                return;
        }
        else {
                *path='L';
                (*pathlen)++;
                insert_tree(&((*n)->l),word,path+1,pathlen);
                return;
        }
}

int main()
{
        FILE *fp,*fout;
        char current[12],path[120];
        char result[100][120];
        int count,I;
        int maxpath,pathlen,sumpath;
        Tnode *root;

        fp=fopen("INPUT3.TXT","rt");
        fout=fopen("OUTPUT3.TXT","wt");

        maxpath=sumpath=0;
        root=NULL;

        fscanf(fp,"%d\n",&count);

        for(I=0;I<count;I++) {
                char buf_[80];
                fgets(buf_,80,fp);
                sscanf(buf_,"%s",current);
                pathlen=0;
                insert_tree(&root,current,path,&pathlen);

                switch (pathlen) {
                        case -1:
```



```
                                strcpy(result[I],"EXISTS");
                                break;
                        case 0:
                                strcpy(result[I],"ROOT");
                                break;
                        default:
                                /* printf("%d\n",pathlen);
                                printf("%s\n",path); */
                                strcpy(result[I],path);
                                sumpath+=pathlen;
                                if (pathlen>maxpath)
                                        maxpath=pathlen;
                                break;
                }
        }

        fprintf(fout,"%d %d\n",maxpath,sumpath);
        for(I=0;I<count;I++) {
                fprintf(fout,"%s\n",result[I]);
        }
        fclose(fout);
        fclose(fp);
        return 0;
}
```