功能实现介绍

```
/*题目1：编写递归函数char *itostr (int n,char *string)，
 *该函数将整数n转换为十进制表示的字符串。（提示：使用递归方法）*/
/*函数功能：将整数n转化成字符串返回
 * string应该提前分配足够长的内存*/
char *itostr(int n,char *string);
/*将整数n（不超过整数-2^31~2^31-1范围）转换成字符串返回*/
char *itostr(long long n);//用于递归

/*题目2：编码实现字符串类CNString*/
class CNString{
private:
    char *str;//指向字符串第一个字符
    int len;//字符串长度（不包含字符'\0'）
public:
    /*默认构造函数*/
    CNString();
    /*构造函数*/
    CNString(const char*st);
    /*复制构造函数*/
    CNString(const CNString &s);

    /*字符串复制函数，将字符串s2复制到s1，
     * 并返回指向该字符串的第一个字符的指针
     * 字符串s1必须足够长*/
    char *strcpy_s(char*s1,const char *s2);

    /*重载=操作*/
    CNString& operator=(const CNString &s);
    /*重载+操作*/
    CNString operator+(const CNString &s);
    char operator[](const int n);
    bool operator<(const CNString &s);
    bool operator>(const CNString &s);
    bool operator==(const CNString &s);

    /*重载输出<<操作
     * 在输出流os中输出字符串s，
     * 并返回输出流*/
    friend std::ostream& operator<<(std::ostream & os,const CNString &s);

    /*析构函数*/
    ~CNString();
};



//题目3：创建双向链表类
struct dNode{
    int key;
    dNode * pre;
    dNode * next;
    dNode():key(0),pre(NULL),next(NULL){}
    dNode(int nkey):key(nkey),pre(NULL),next(NULL){}
```

```
};//定义双向链表的节点

class dlist{
private:
    dNode * head;//链表头
    dNode * tail;//链表尾
    int len;//链表长度
public:
    dlist();
    dlist(dlist & dl);

    /*在双向链表末尾插入节点Node*/
    void insert(dNode & Node);

    /*在双向链表末尾插入以nkey为key的节点*/
    void insert(int nkey);

    /*链表排序，默认为升序（index = 0）
     * 参数index = 1，则降序*/
    void sort(const int index = 0);

    /*查找链表中是否含有节点Node
     * 若找到，返回节点Node在链表中第一次出现的位置
     * 若没有找到，返回0*/
    int search(dNode & Node);

    /*删除链表中key为nk的一个节点，并返回true
     * 若删除失败，则返回false*/
    bool del(int nk);

    /*删除链表中与Node数据项key相同的一个节点
     * 并返回true,否则返回false*/
    bool del(dNode & Node);

    /*打印链表中节点的数据项*/
    void printNode() const;
    ~dlist();
};

//题目4：创建单向链表类
struct sNode{
    int key;
    sNode *next;
    sNode():key(0),next(NULL){}
    sNode(int nkey):key(nkey),next(NULL){}
};
class slist{
private:
    sNode *head;
    int len;
public:
    slist();

    slist(slist & sl);
```

```
    /*在链表末尾插入以nkey为key的节点*/
    void insert(int nk);

    /*在链表末尾插入节点Node*/
    void insert(sNode & Node);

    /*查找链表中是否含有节点Node
     * 若找到，返回节点Node在链表中第一次出现的位置
     * 若没有找到，返回0*/
    int search(sNode & Node);

    /*删除链表中key为nk的一个节点，并返回true
     * 若删除失败，则返回false*/
    void del(int nk);

    /*删除链表中与Node数据项key相同的一个节点
     * 并返回true,否则返回false*/
    void del(sNode & Node);

    /*链表排序，默认为升序（index = 0）
     * 参数index = 1，则降序*/
    void sort(const int index = 0);

    /*打印链表中节点的数据项*/
    void printNode() const;

    ~slist();
};
```