by 黄万平；Codeforces ID: hwpvector

需要注意的是，如果一个类里面有指针，最好要实现以下类特殊成员函数（使用 STL 库相关算法和容器时一定要实现）：

- 深拷贝构造函数
    - `T (const T &s) {将 s 和 s 元素中指针所指的元素 拷贝过来}`
- 重载深拷贝的 =
    - `string& operator = (const string &s) {析构 *this；将 s 和 s 元素中指针所指的元素 拷贝过来}`
    - 需要注意，在开始时，需要判断有没有自己给自己赋值，不然析构后拷贝会有问题；具体地： `if (this==&s) return *this;`
- 析构函数
    - 析构指针元素

原因：如果只有析构函数，会产生以下情况

```
T s1; T s2(s1); T s2=s1; // 这两种情况调用的是构造函数
T s1; T s2; s2=s1; // 调用重载的=
```

默认的拷贝构造函数和重载=是直接将指针复制过来了，没有重新拷贝一份指针所指向的元素。析构时会 double free

还需要注意的是，在一个类里面，如果写了其他的构造函数，最好把默认构造函数也写一下，说不定什么 STL 容器/算法里就会调用默认构造函数。

请勿抄袭。仅供参考。如果哪题有发现内存泄漏之类的，请私信我！

Problem 1

```
#include <bits/stdc++.h>

using std::cin; using std::cout; using std::flush;

class Time {
private:
    int hour,minute,second;
public:
    Time (int _h=0,int _m=0,int _s=0) : hour(_h),minute(_m),second(_s) {}
    std::string to_string() const {
```

```cpp
        auto _to_string=[](const int &num) {
            if (num<10) return "0"+std::to_string(num);
            return std::to_string(num);
        };
        return _to_string(hour)+":"+_to_string(minute)+":"+_to_string(secon
    }
    friend std::istream& operator >> (std::istream &in,Time &t) {
        in>>t.hour>>t.minute>>t.second;
        return in;
    }
    friend std::ostream& operator << (std::ostream &out,const Time &t) {
        out<<t.to_string()<<'\n';
        return out;
    }
    Time operator + (const Time &t) const {
        Time ans={hour+t.hour,minute+t.minute,second+t.second};
        if (ans.second>=60) ans.second-=60,ans.minute+=1;
        if (ans.minute>=60) ans.minute-=60,ans.hour+=1;
        if (ans.hour>=24) ans.hour-=24;
        return ans;
    }
    Time& operator += (const Time &t) {
        return *this=*this+t;
    }
    Time operator - (const Time &t) const {
        Time ans={hour-t.hour,minute-t.minute,second-t.second};
        if (ans.second<0) ans.second+=60,ans.minute-=1;
        if (ans.minute<0) ans.minute+=60,ans.hour-=1;
        if (ans.hour<0) ans.hour+=24;
        return ans;
    }
    Time& operator -= (const Time &t) {
        return *this=*this-t;
    }
    Time& operator ++ () {
        return *this=*this+Time(0,0,1);
    }
    Time operator ++ (int) {
        Time t=*this; *this=*this+Time(0,0,1);
        return t;
    }
    Time& operator -- () {
        return *this=*this-Time(0,0,1);
    }
    Time operator -- (int) {
        Time t=*this; *this=*this-Time(0,0,1);
        return t;
    }
};

int main() {
    Time T1,T2;
    cin>>T1>>T2;
    cout<<(T1+=(T2++));
    cout<<(T1-=T2);
    cout<<(++T2);
    cout<<(T2+=(T1--));
    cout<<(--T1);
    cout<<(T2-=T1);
    return 0;
}
```

Problem 2

```
#include <bits/stdc++.h>

using std::cin; using std::cout; using std::flush;

class strfind {
    std::string str;
public:
    void input() {
        cin>>str;
    }
    std::vector<size_t> find(const char &c) const {
        int cnt=0;
        std::vector<size_t> pos;
        for (auto s:str) {
            if (c==s) pos.push_back(cnt);
            ++cnt;
        }
        return pos;
    }
};

int main() {
    strfind s; s.input();
    char c; cin>>c;
    std::vector<size_t> ans=s.find(c);
    if (ans.size()==0) {
        cout<<"NULL\n";
    }
    else {
        for (auto v:ans) {
            cout<<v<<' ';
        }
        cout<<'\n';
    }
    return 0;
}
```

Problem 3

```
#include <bits/stdc++.h>

using std::cin; using std::cout; using std::flush;

class string {
private:
    char *str;
    size_t len;
    size_t cap;
public:
    string(const char *s="") {
        len=strlen(s); cap=len;
        str=new char[len+1];
        strcpy(str,s);
    }
    string(const string &s) {
        len=s.len; cap=len;
        str=new char[len+1];
        strcpy(str,s.str);
    }
    ~string() {
        delete []str;
    }
    string& operator = (const string &s) {
        if (this!=&s) {
            delete []str;
            len=s.len; cap=s.cap;
            str=new char[len+1];
            strcpy(str,s.str);
        }
        return *this;
    }
    void clear() {
        delete []str;
        str=new char[1];
        str[0]='\0';
        len=cap=0;
    }
    void reserve(size_t n) {
        char *s=new char[n+1];
        strcpy(s,str);
        delete []str;
        str=s; cap=n;
    }
    void push_back(char ch) {
        ++len;
        if (len>cap) reserve(cap==0?4:cap*2);
        str[len-1]=ch; str[len]='\0';
    }
    friend std::istream& operator >> (std::istream& in,string &s) {
        s.clear();
        char ch=getchar();
        while (ch==' '||ch=='\n'||ch=='\r') ch=getchar();
        while (ch!=' '&&ch!='\n'&&ch!='\r') s.push_back(ch),ch=getchar();
        return in;
    }
    friend std::ostream& operator << (std::ostream& out,const string &s) {
        out<<s.str;
        return out;
    }
    void del(const char &c) {
        size_t cnt=0;
        char *newstr;
        for (size_t i=0;i<len;i++)
            if (c!=str[i]) ++cnt;
        newstr=new char[cnt+1]; newstr[cnt]='\0';
        for (size_t i=0,j=0;i<len;i++)
            if (c!=str[i]) newstr[j]=str[i],++j;
        len=cnt; delete []str; str=newstr;
    }
    int checkend() {
        if (len==1&&str[0]=='@') return 1;
        return 0;
    }
    bool operator < (const string &s) const {
```

```cpp
        size_t minlen=std::min(s.len,len);
        for (size_t i=0;i<minlen;i++) {
            if (str[i]!=s.str[i])
                return str[i]<s.str[i];
        }
        return len<s.len;
    }
};

int main() {
    char c; cin>>c;
    std::vector<string> v;
    string s;
    while (cin>>s) {
        if (s.checkend()) break;
        s.del(c); v.push_back(s);
    }
    std::sort(v.begin(),v.end());
    for (auto it=v.rbegin();it!=v.rend();++it) {
        cout<<(*it)<<'\n';
    }
    return 0;
}
```

Problem 4

```cpp
#include <bits/stdc++.h>

using std::cin; using std::cout; using std::flush;

class string {
private:
    char *str;
    size_t len;
    size_t cap;
public:
    string(const char *s="") {
        len=strlen(s); cap=len;
        str=new char[len+1];
        strcpy(str,s);
    }
    string(const string &s) {
        len=s.len; cap=len;
        str=new char[len+1];
        strcpy(str,s.str);
    }
    ~string() {
        delete []str;
    }
    string& operator = (const string &s) {
        if (this!=&s) {
            delete []str;
            len=s.len; cap=s.cap;
            str=new char[len+1];
            strcpy(str,s.str);
        }
        return *this;
    }
    void clear() {
        delete []str;
        str=new char[1];
        str[0]='\0';
        len=cap=0;
    }
    void reserve(size_t n) {
        char *s=new char[n+1];
        strcpy(s,str);
        delete []str;
        str=s; cap=n;
    }
    void push_back(char ch) {
        ++len;
        if (len>cap) reserve(cap==0?4:cap*2);
        str[len-1]=ch; str[len]='\0';
    }
    friend std::istream& operator >> (std::istream& in,string &s) {
        s.clear();
        char ch=getchar();
        while (ch==' '||ch=='\n'||ch=='\r') ch=getchar();
        while (ch!=' '&&ch!='\n'&&ch!='\r') s.push_back(ch),ch=getchar();
        return in;
    }
    friend std::ostream& operator << (std::ostream& out,const string &s) {
        out<<s.str;
        return out;
    }
    bool operator < (const string &s) const {
        size_t minlen=std::min(s.len,len);
        for (size_t i=0;i<minlen;i++) {
                if (str[i]!=s.str[i])
                    return str[i]<s.str[i];
            }
            return len<s.len;
        }
        string& operator += (const string &s) {
            while (len+s.len>cap) reserve(cap==0?4:cap*2);
            for (size_t i=0;i<s.len;i++) str[len]=s.str[i],++len;
            str[len]='\0';
            return *this;
        }
        string operator + (const string &s) const {
            string ans=*this; ans+=s;
            return ans;
        }
        string& zRearrange(int n) {
            if (n==1) return *this;
            std::vector<string> v(n);
            int w=1;
            for (size_t i=0,j=0;i<len;i++,j+=w) {
                v[j].push_back(str[i]);
                if (j==0) w=1;
                else if (j==n-1) w=-1;
            }
            clear();
            for (int i=0;i<n;i++) *this+=v[i];
            return *this;
        }
};

int main() {
    int n; cin>>n;
    string s; cin>>s;
    cout<<s.zRearrange(n);
    return 0;
}
```

Problem 5

```
#include <bits/stdc++.h>

using std::cin; using std::cout; using std::flush;
```

```
class Int {
private:
    int x;
public:
    Int(int _x=0) : x(_x) {}
    std::string encode() {
        std::string ans;
        while (x>0) {
            ans.push_back(x%10+13-1+'A');
            x/=10;
        }
        std::reverse(ans.begin(),ans.end());
        return ans;
    }
};

int main() {
    int n; cin>>n;
    Int m(n);
    cout<<m.encode();
    return 0;
}
```

Problem 6

```
#include <bits/stdc++.h>

using std::cin; using std::cout; using std::flush;

class Matrix {
private:
    double **mat;
    int row,col;
public:
    Matrix() {
        row=col=0;
        mat=nullptr;
    }
    Matrix(const int &_r,const int &_c) {
        row=_r,col=_c;
        mat=new double*[row];
        for (int i=0;i<row;i++)
            mat[i]=new double[col]();
```

```
}
Matrix(const Matrix& m) {
    row=m.row,col=m.col;
    mat=new double*[row];
    for (int i=0;i<row;i++) {
        mat[i]=new double[col];
        memcpy(mat[i],m.mat[i],col*sizeof(double));
    }
}
~Matrix() {
    for (int i=0;i<row;i++)
        delete []mat[i];
    delete []mat;
}
Matrix& operator = (const Matrix& m) {
    if (this!=&m) {
        for (int i=0;i<row;i++)
            delete []mat[i];
        delete []mat;
        row=m.row,col=m.col;
        mat=new double*[row];
        for (int i=0;i<row;i++) {
            mat[i]=new double[col];
            memcpy(mat[i],m.mat[i],col*sizeof(double));
        }
    }
    return *this;
}
double& operator () (int i,int j) {
    if (!(i<row&&j<col)) throw -1;
    return mat[i][j];
}
const double& operator () (int i,int j) const {
    if (!(i<row&&j<col)) throw -1;
    return mat[i][j];
}
friend std::istream& operator >> (std::istream &in,Matrix &m) {
    for (int i=0;i<m.row;i++) {
        for (int j=0;j<m.col;j++) in>>m(i,j);
    }
    return in;
}
friend std::ostream& operator << (std::ostream &out,const Matrix &m) {
    for (int i=0;i<m.row;i++) {
        for (int j=0;j<m.col;j++) out<<m(i,j)<<' ';
        out<<'\n';
        }
        return out;
    }
    Matrix& operator += (const Matrix &m) {
        if (!(row==m.row&&col==m.col)) throw -2;
        for (int i=0;i<row;i++) {
            for (int j=0;j<m.col;j++) mat[i][j]+=m(i,j);
        }
        return *this;
    }
    Matrix& operator *= (const Matrix &m) {
        if (!(col==m.row)) throw -2;
        Matrix ans(row,m.col);
        for (int i=0;i<row;i++) {
            for (int k=0;k<col;k++){
                for (int j=0;j<m.col;j++)
                    ans(i,j)+=mat[i][k]*m(k,j);
            }
        }
        return *this=ans;
    }
};

int main() {
    int row1,col1,row2,col2;
    cin>>row1>>col1;
    Matrix table1(row1,col1);
    cin>>table1;
    cin>>row2>>col2;
    Matrix table2(row2,col2);
    cin>>table2;
    try {cout<<table1(row1/2,col1/2)<<'\n';}
    catch(int op) {cout<<"ERROR!\n";}
    try {cout<<(table1*=table2);}
    catch(int op) {cout<<"ERROR!\n";}
    try {cout<<(table1+=table2);}
    catch(int op) {cout<<"ERROR!\n";}
    try {cout<<(table1=table2);}
    catch(int op) {cout<<"ERROR!\n";}
    return 0;
}
```

Problem 7

```
#include <bits/stdc++.h>

using std::cin; using std::cout; using std::flush;

class score {
private:
    std::string name;
    int s1,s2,s3,sum;
public:
    friend std::istream& operator >> (std::istream &in,score &s) {
        in>>s.name>>s.s1>>s.s2>>s.s3;
        s.sum=s.s1+s.s2+s.s3;
        return in;
    }
    friend std::ostream& operator << (std::ostream &out,const score &s) {
        out<<s.name<<' '<<s.sum;
        return out;
    }
    bool operator < (const score &s) const {
        if (sum!=s.sum) return sum>s.sum;
        if (s1!=s.s1) return s1>s.s1;
        if (s2!=s.s2) return s2>s.s2;
        if (s3!=s.s3) return s3>s.s3;
        return name>s.name;
    }
};

int main() {
    int n; cin>>n;
    std::vector<score> v(n);
    for (int i=0;i<n;i++)
        cin>>v[i];
    int k; cin>>k;
    std::sort(v.begin(),v.end());
    cout<<v[k-1]<<'\n';
    return 0;
}
```

Problem 8

```
#include <bits/stdc++.h>

using std::cin; using std::cout; using std::flush;
```

```cpp
struct node {
    int val;
    node *nxt;
};
class Joseph {
private:
    int n,m;
    node *head;
public:
    Joseph() {
        n=m=0;
        head=nullptr;
    }
    Joseph(int _n,int _m) : n(_n), m(_m) {
        head=new node; head->val=1;
        node *to=head;
        for (int i=2;i<=n;i++) {
            to->nxt=new node;
            to=to->nxt; to->val=i;
        }
        to->nxt=head; head=to;
    }
    ~Joseph() {
        if (head!=nullptr) {
            while (head->nxt!=head) {
                node *del=head->nxt;
                head->nxt=head->nxt->nxt;
                delete del;
            }
            delete head;
        }
    }
    Joseph(const Joseph &J)=delete;
    Joseph& operator = (const Joseph &J) const=delete;
    void simulate(int k) {
        for (int i=2;i<=k;i++) head=head->nxt;
        for (int i=1;i<=n;i++) {
            for (int i=1;i<m;i++) head=head->nxt;
            node *del=head->nxt; head->nxt=head->nxt->nxt;
            cout<<del->val<<' '<<flush;
            delete del;
        }
        head=nullptr;
    }
};
```

```
int main() {
    int n,m,k;
    cin>>n>>m>>k;
    Joseph obj(n,m);
    obj.simulate(k);
    return 0;
}
```

Problem 9

```
#include <bits/stdc++.h>

using std::cin; using std::cout; using std::flush;

class Cow {
private:
    char name[20];
    char *hobby;
    double weight;
public:
    Cow() {
        name[0]='\0';
        hobby=new char[1];
        hobby[0]='\0';
        weight=0;
    }
    Cow(const char *nm,const char *ho,double wt) {
        strcpy(name,nm);
        hobby=new char[strlen(ho)+1];
        strcpy(hobby,ho);
        weight=wt;
    }
    Cow(const Cow &c) {
        strcpy(name,c.name);
        hobby=new char[strlen(c.hobby)+1];
        strcpy(hobby,c.hobby);
        weight=c.weight;
    }
    ~Cow() {
        delete []hobby;
    }
    Cow& operator = (const Cow &c) {
```

```cpp
        if (this!=&c) {
            delete []hobby;
            strcpy(name,c.name);
            hobby=new char[strlen(c.hobby)+1];
            strcpy(hobby,c.hobby);
            weight=c.weight;
        }
        return *this;
    }
    void ShowCow() const {
        cout<<name<<' '<<hobby<<' '<<weight<<'\n';
    }
};

int main() {
    char *name=new char[1024],*hobby=new char[1024];
    double weight;
    cin.getline(name,1024);
    cin.getline(hobby,1024);
    cin>>weight;
    Cow cow1(name,hobby,weight);
    cin.ignore();
    cin.getline(name,1024);
    cin.getline(hobby,1024);
    cin>>weight;
    Cow cow2(name,hobby,weight);
    delete []name; delete []hobby;
    Cow cow3(cow1);
    Cow cow4;
    cow4=cow2;
    cow3.ShowCow();
    cow4.ShowCow();
    return 0;
}
```