```
/*-------------CLASS MyString----------------------*/
#include<iostream.h>
const int n =20;
class MyString
{
friend ostream& operator<<(ostream& o, const MyString& m);
//Funziona solo se la stringa letta ha dimensione inferiore a m.length
friend istream& operator>>(istream& i, MyString& m);
    public:
        MyString();
        MyString(const char c[]);
        MyString(const MyString&);
        char getCharAt(unsigned int i) ;
        void setCharAt(unsigned int i, char c);
        bool append(const MyString& s);
        int getLength() ;

        MyString& operator=(const MyString& m);
        bool operator==(const MyString& m);
        bool operator<=(const MyString& m);
        bool operator<(const MyString& m) ;
        bool operator>=(const MyString& m);
        bool operator>(const MyString& m) ;
        char& operator[](unsigned int) ;
        MyString& operator+=(const MyString& m);


    private:
        char str[n];
        unsigned int length;
};

MyString::MyString() : length(0)
{
    str[0] = '\0';
}

MyString::MyString(const char c[])
{
    length = strlen(c);
    strcpy(str,c);
}

MyString::MyString(const MyString& s)
{
    length = s.length;
    strcpy(str,s.str);
}

char MyString::getCharAt(unsigned int i)
{
    return str[i-1];
}

void MyString::setCharAt(unsigned int i, char c)
{
    str[i-1]=c;
}

bool MyString::append(const MyString& s)
{
    if(length+s.length<n)
    {
```

```
        strcat(str, s.str);
        length += s.length;
        return true;
    }
    return false;
}

int MyString::getLength()
{
    return length;
}

MyString& MyString::operator=(const MyString& m)
{
    strcpy(str,m.str);
    return *this;
}

bool MyString::operator==(const MyString& m)
{
    return strcmp(str,m.str) == 0;
}

bool MyString::operator<=(const MyString& m)
{
    return strcmp(str,m.str) <= 0;
}

bool MyString::operator<(const MyString& m)
{
    return strcmp(str,m.str) < 0;
}

bool MyString::operator>=(const MyString& m)
{
    return strcmp(str,m.str) >= 0;
}

bool MyString::operator>(const MyString& m)
{
    return strcmp(str,m.str) > 0;
}

char& MyString::operator[](unsigned int i)
{
    return str[i-1];
}

MyString& MyString::operator+=(const MyString& m)
{
    append(m);
    return *this;
}

ostream& operator<<(ostream& o, const MyString& m)
{
    o << m.str;
    return o;
}


istream& operator>>(istream& i,  MyString& m)
{
    i >> m.str;
```

```
    return i;
}



//----------->CLASSE REGALO<----------------------------
class Regalo{
    friend istream& operator>>(istream&, Regalo& );
    friend ostream& operator<<(ostream&, Regalo& );
private:
    MyString nome;
    int tipo;
    float prezzo;

public:
    bool operator==(Regalo r);
    bool operator>(Regalo r);
    int restituisciTipo(){return tipo;}
    float restituisciPrezzo(){return prezzo;}
};

bool Regalo::operator ==(Regalo r){

    if ((r.nome==nome)&&(r.tipo==tipo)&&(r.prezzo==prezzo))
        return true;
    return false;
}

bool Regalo::operator >(Regalo r){
    if(prezzo>r.prezzo)
        return true;
    return false;
}

istream& operator>>(istream& in, Regalo& r){
    cout<<"Inserisci tipo "<<endl;
    cout<<"  (1: dolcini, 2: vestiario, 3:giochi) ";
    in>>r.tipo;
    cout<<"  Inserisci nome ";
    in>>r.nome;
    cout<<"  Inserisci prezzo ";
    in>>r.prezzo;
    return in;
}

ostream& operator<<(ostream& o, Regalo& r){
    o<<r.nome<<", "<<r.prezzo<<" euro";
    return o;
}
////////////////////////////////////////////////////////
//----------->CLASSE CALZA<----------------------------
const int dim=100;
class Calza{
    friend istream& operator>>(istream&, Calza& );
    friend ostream& operator<<(ostream&, Calza& );
private:
    Regalo dolci[dim];
    Regalo vestiti[dim];
    Regalo giochi[dim];
    int dimD;
    int dimV;
    int dimG;
public:
    bool esisteTipo(int);
```

```
    int quantiDiTipo(int);
    float valoreTipo(int);
    float valoreCalza();
    void inserisciElemento();
    bool ordinata(int, bool, bool, bool);
};

bool Calza::esisteTipo(int t){
    switch(t)
    {
    case 1:
        if(dimD > 0)
            return true;
        break;
    case 2:
        if(dimV > 0)
            return true;
        break;
    case 3:
        if(dimG > 0)
            return true;
        break;
    }
    return false;
}

int Calza::quantiDiTipo(int t){
    switch(t)
    {
    case 1:
        return dimD;
        break;
    case 2:
        return dimV;
        break;
    case 3:
        return dimG;
        break;
    }
    return 0;
}

float Calza::valoreTipo(int t){
    float costoTot=0;
    switch(t)
    {
    case 1:
        for(int i=0; i<dimD; i++)
            costoTot+= dolci[i].restituisciPrezzo();
        break;
    case 2:
        for(int j=0; j<dimV; j++)
            costoTot+= vestiti[j].restituisciPrezzo();
        break;
    case 3:
        for(int k=0; k<dimG; k++)
            costoTot+= giochi[k].restituisciPrezzo();
        break;
    }
    return costoTot;
}

float Calza::valoreCalza(){
```

```
    float costoTot=0;
    for(int i=0; i<dimD; i++)
        costoTot+= dolci[i].restituisciPrezzo();

    for(int j=0; j<dimV; j++)
        costoTot+= vestiti[j].restituisciPrezzo();

    for(int k=0; k<dimG; k++)
        costoTot+= giochi[k].restituisciPrezzo();

    return costoTot;
}

bool Calza::ordinata(int inizio, bool ordD, bool ordV, bool ordG){

    if(ordD && ordV && ordG)
        return true;

    if( inizio==dimD )
        ordD = true;
    if( inizio==dimG )
        ordG=true;
    if( inizio==dimV )
        ordV=true;

    if((!ordD) && (dolci[inizio+1] > dolci[inizio]))
        return false;
    if((!ordV) && (vestiti[inizio+1] > vestiti[inizio]))
        return false;
    if((!ordG) && (giochi[inizio+1] > giochi[inizio]))
        return false;

    return ordinata(inizio+1,ordD,ordV,ordG);
}

istream& operator>>(istream& in, Calza& c ){
    cout<<"Inserisci numero di Dolcini(<100): ";
    in>>c.dimD;
    cout<<"Inserisci i Dolcini: ";
    for(int i=0; i<c.dimD; i++)
        in>>c.dolci[i];

    cout<<"Inserisci numero di Vestiti(<100): ";
    in>>c.dimV;
    cout<<"Inserisci i Vestiti: ";
    for(int i1=0; i1<c.dimV; i1++)
        in>>c.vestiti[i1];

    cout<<"Inserisci numero di Giochi(<100): ";
    in>>c.dimG;
    cout<<"Inserisci i Giochi: ";
    for(int i2=0; i2<c.dimG; i2++)
        in>>c.giochi[i2];

    return in;
}

ostream& operator<<(ostream& o, Calza& c){
    o<<"Dolcini: "<<endl;
    for(int j=0; j<c.dimD; j++)
        o<<c.dolci[j];

    o<<"Vestiario: "<<endl;
    for(int i1=0; i1<c.dimV; i1++)
```

```
        o<<c.vestiti[i1];

    o<<"Giochi: "<<endl;
    for(int i2=0; i2<c.dimG; i2++)
        o<<c.giochi[i2];

    return o;
}



////////////////////////////////////////////////
int main()
{
    Calza c;
    float budget;
    cin>>c;
    cout<<"Inserisci budget massimo: ";
    cin>>budget;

    if(!c.esisteTipo(1))
        cout<<"Nella calza mancano i regali per la categoria dolcini";
    if(!c.esisteTipo(2))
        cout<<"Nella calza mancano i regali per la categoria vestiti";
    if(!c.esisteTipo(3))
        cout<<"Nella calza mancano i regali per la categoria giocattoli";

    int numDolci = c.quantiDiTipo(1);
    int numVest = c.quantiDiTipo(2);
    int numGiochi = c.quantiDiTipo(3);
    if(numDolci > numVest || numDolci > numGiochi)
        cout<<"La calza contiene troppo dolci!"<<endl;

    if(c.valoreCalza() > budget)
        cout<<"La calza è fuori budget!"<<endl;

    if(c.valoreTipo(3)>c.valoreTipo(2) && c.valoreTipo(3)>c.valoreTipo(1))
        cout<<"La somma spesa in giochi è maggiore delle altre."<<endl;
    else
        cout<<"La somma spesa in giochi non è maggiore delle altre."<<endl;

    if(c.ordinata(0,false,false,false))
        cout<<"la calza è ordinata!"<<endl;
    return 0;

}
```