/* PROBLEM STATEMENT

--Imagine a publishing company which does marketing for book and audiocassette versions. Create a class publication that stores the title (a string) and price (type float) of a publication.

From this class derive two classes: book, which adds a page count (type int), and tape, which adds a playing time in minutes (type float).

Write a program that instantiates the book and tape classes, allows user to enter data and displays the data members. If an exception is caught, replace all the data member values with zero values.

***/**

```
#include<iostream>
#include<string.h>

using namespace std;

class Publication
{
  protected:
  string title;
  float price;
  public:
   void Set_Zero(void) // to set values to zero to handel exception..
   {
     title="000";
     price = 0.0;
   }

   void getdata(void)
   {
     cout<<"\n\t Enter the Name: ";
     cin>>title;
     cout<<"\t Enter the price: ";
     cin>>price;
   }

   void putdata(void)
   {
     cout<<"\n\t Name: "<<title;
     cout<<"\t Price: "<<price;
   }
};

class Book : public Publication
{
```

```
  int page_count;
  public:
   void Set_Zero(void) // to set values to zero to handel exception..
   {
     page_count=0;
     Publication::Set_Zero();
   }

   void getdata(void)
    {
     Publication::getdata();
     cout<<"\t Enter the number of pages: ";
     cin>>page_count;
     try
      {

        if(price<0)
         throw (0);  // throwing an integer type exception.. to show,
exception caught in price
        if(page_count<0)
         throw (0.0); // throwing a double type exception.. to show,
exception caught in page_count

      }
       catch(double x)
      {
        cout<<"\n\t Exception-- Page count cannot hold a negetive
value:";
     cout<<"\n\t Turning all values to zero.";
         Set_Zero();

      }
      catch(int x)
      {
        cout<<"\n\t Exception-- Price cannot hold a negetive value:";
     cout<<"\n\t Turning all values to zero.";
         Set_Zero();
      }

   }

   void putdata(void)
   {
     Publication::putdata();
     cout<<"\t Number of pages: "<<page_count;
     cout<<endl<<endl;
   }
};
```

```
class Tape : public Publication
{
  float playing_time;
  public:
   void Set_Zero(void)                              // to set values to
zero to handel exception..
   {
     playing_time=0.0;
     Publication::Set_Zero();
   }

   void getdata(void)
    {
     Publication::getdata();
     cout<<"\t Enter the plying time: ";
     cin>>playing_time;
     try
      {

        if(price<0)
         throw 0;                                    // throwing an
integer type exception.. to show, exception caught in price

        if(playing_time<0)
        throw (0.0);                                 // throwing an
double type exception.. to show, exception caught in playing_time

      }
      catch(int x)
      {
        cout<<"\n\t Exception-- Price cannot be a negetive value:";
     cout<<"\n\t Turning all values to zero.";
         Set_Zero();
      }
      catch(double x)
      {
        cout<<"\n\t Exception-- Playing time cannot be a negetive
value:";
     cout<<"\n\t Turning all values to zero.";
         Set_Zero();
      }
   }

   void putdata(void)
   {
     Publication::putdata();
```

```
     cout<<"\t Plying time: "<<playing_time;
     cout<<endl<<endl;
   }
};

int main()
{
 Tape T;
 Book B;
 cout<<"\n\t Enter data of Tape :\n";
 T.getdata();
 T.putdata();
 cout<<"\n\t Enter data of Book :\n";
 B.getdata();
 B.putdata();
 return 0;
}
```