```
#Input:
#include<iostream>
using namespace std;
class publication
{
  public:
  string title;
  float price;
  void add()
  {
    cout<<"Enter title: ";
   // getline(cin,title);
    cin>>title;
    cout<<"Enter price: ";
    cin>>price;
  }
  void display()
  {
    cout<<"Name of book is: "<<title<<endl;
    if(price<0)
    {
      cout<<"Invalid price!!!\n";
    }
    else
      cout<<"Price of book is: "<<price<<endl;
  }
};
class PageCount: public publication
{
  public:
  int pgct;
  void pagecount()
  {
    cout<<"Number of pages: ";
    cin>>pgct;
  }
  void add1()
  {
    add();
  }
  void display1()
  {
    display();
  }
  void display_pgcount()
  {
    cout<<"Pagecount is: "<<pgct<<endl;
  }
};
class tape: public publication
{
  public:
```

```
    int pltm;
    void playtime()
    {
        cout<<"Enter playtime: ";
        cin>>pltm;
    }
    void display_tape()
    {
        if(pltm<0)
        {
            cout<<"Invalid playtime!!!\n";
        }
        else
            cout<<"Duration is: "<<pltm<<" minutes"<<endl;
    }
};
int main()
{
    PageCount obj;
    tape obj1;
    int flag = 1;
    int ch = 0;
    while(flag)
    {
        cout<<endl<<endl;
        cout<<"YOUR CHOICES ARE!!!\n";
        cout<<"1.Add book title and price: \n";
        cout<<"2.Add book page count: \n";
        cout<<"3.Add duration: \n";
        cout<<"4.Display book title and price: \n";
        cout<<"5.Display pagecount: \n";
        cout<<"6.Display duration: \n";
        cout<<"7.Exit\n";
        cout<<"Enter choice: \n";
        cin>>ch;
        switch(ch)
        {
            case 1:
            obj.add1();
            break;

            case 2:
            obj.pagecount();
            break;

            case 3:
            obj1.playtime();
            break;

            case 4:
            obj.display1();
            break;
```

```
            case 5:
            obj.display_pgcount();
            break;

            case 6:
            obj1.display_tape();
            break;

            case 7:
            flag = 0;
            break;

            default:
            cout<<"Invalid choice!!!\n";
            break;
        }
    }
    return 0;
}
```

Q1. How do the properties of the following two derived classes differ?

a. class D1: private B{//...};

b. class D2: public B{//...};

class D1: private B{//...}	class D2: public B{//...}
1. Private members of base class B are not accessible in derived class D1	1. Private members of base class B are not accessible in Derived class D2.
2. Protected members of base class B become private members of Derived class D1.	2. Protected members of base class B ~~become~~ remain protected in Derived class D2.
3. Public members of base class B become private members of Derived class D1.	3. Public members of Base class B remain public in Derived class D2.
4. So other classes can use public members of base class through Derived class object.	4. So, no members of Base class can be accessed by other classes through Derived class object as they are private in Derived class. So, even subclass of Derived class can't access them.

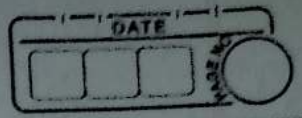
Q2. When do we use the protected visibility specifier to a class member?

Ans.
- Protected variables allow access to the variables only from sub-classes and classes within the same package.
- Protected variables can be useful if you want your data to be read-only, or when you want to abstract your data.
- We use protected access specifier on the members of a class when we want to access them from the child class of the same object.

Q3. Describe the syntax of single inheritance in c++.

Ans.
```
class Base_Class
{
    //class defination
};
class Derived_Class:access_specifiers Base_Class
{
    //class defination
};
int main()
{
    Base_Class obj1;
    Derived_Class obj2;
    return 0;
}
```

→ Here in the above syntax, Base_Class is the name of



the Base class.

- Derived_class is the name of the Derived-class, and access_specifiers specifies how the derived class will inherit the members of the base class.
- access-specifiers can be private, protected or public.