Software Development Fundamentals – II
Tutorial 13 Solution

Q1)
The access specifiers that can prevent from class members being inherited are:
> Private
> Protected

Q2)
All three can be used together in the class.

Q3)
The add() function of class A

Q4)
It will give error as private and protected data members cannot be accessed by the object declared in the main function.

Q5)
The output will be:
Protected: 20
Private: 30

Q6)
The output will be:
This is Base class
This is Base class

Q7)
The output will be:
5

Q8)
>Single Inheritance: In single inheritance, a class is allowed to inherit from only one class. i.e. one sub class is inherited by one base class only.
Example:

```
#include <iostream>
using namespace std;
class Vehicle {
  public:
    Vehicle()
    {
      cout << "This is a Vehicle" << endl;
    }
};
class Car: public Vehicle{

};
int main()
{
    Car obj;
    return 0;
}
```

Output:
This is a vehicle

>Multiple Inheritance: Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. i.e one sub class is inherited from more than one base classes.

Example:

```
  #include <iostream>
  using namespace std;
  class Vehicle {
   public:
    Vehicle()
    {
     cout << "This is a Vehicle" << endl;
    }
  };
  class FourWheeler {
   public:
    FourWheeler()
    {
     cout << "This is a 4 wheeler Vehicle" << endl;
    }
  };
  class Car: public Vehicle, public FourWheeler {

  };
  int main()
  {
    Car obj;
    return 0;
```

```
        }
```

Output:
```
This is a Vehicle
This is a 4 wheeler Vehicle
```

>Multilevel Inheritance: In this type of inheritance, a derived class is created from another derived class.

Example:
```
        #include <iostream>
        using namespace std;
        class Vehicle
        {
          public:
            Vehicle()
            {
              cout << "This is a Vehicle" << endl;
            }
        };
        class fourWheeler: public Vehicle
        {  public:
            fourWheeler()
            {
              cout<<"Objects with 4 wheels are vehicles"<<endl;
            }
        };
        class Car: public fourWheeler{
           public:
             car()
             {
               cout<<"Car has 4 Wheels"<<endl;
             }
        };
        int main()
        {
            Car obj;
            return 0;
        }
```

Output:
```
This is a Vehicle
Objects with 4 wheels are vehicles
Car has 4 Wheels
```

>Hierarchical Inheritance: In this type of inheritance, more than one sub class is inherited from a single base class. i.e. more than one derived class is created from a single base class.

Example:
```
#include <iostream>
using namespace std;
class Vehicle
{
  public:
    Vehicle()
    {
      cout << "This is a Vehicle" << endl;
    }
};
class Car: public Vehicle
{

};
class Bus: public Vehicle
{

};
int main()
{
    Car obj1;
    Bus obj2;
    return 0;
}
```
Output:
```
This is a Vehicle
This is a Vehicle
```

>Hybrid (Virtual) Inheritance: Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.
Below image shows the combination of hierarchical and multiple inheritance:
Example:
```
#include <iostream>
using namespace std;
class Vehicle
{
  public:
    Vehicle()
    {
      cout << "This is a Vehicle" << endl;
    }
};
class Fare
{
    public:
    Fare()
    {
        cout<<"Fare of Vehicle\n";
    }
};
class Car: public Vehicle
{

};
class Bus: public Vehicle, public Fare
{

};
int main()
{
    Bus obj2;
    return 0;
}
```

Output:

```
This is a Vehicle
Fare of Vehicle
```

Q9)

We can access the private member of the Base class by using a function inside the base class or by passing the object of the base class in the derived class function of by using friend keyword and making another class friend of base class.

Q10)

```
#include <iostream>
#include <string>
#define min_cur 5000
using namespace std;
class account
{
```

```
public:
   string name;
   long accno;
   char type;
   void input()
   {
      cout << "Enter your name:\n";
      cin >> name;
      cout << "Enter your account number:\n";
      cin >> accno;
      cout << "Enter your account type: 'C' for current, 'S' for savings:\n";
      cin >> type;
   }
};
class cur_accnt : public account
{
public:
   double balance_cur;
   void setv()
   {
      balance_cur = 0.0;
   }
   void deposit()
   {
      cout << "Enter the amount you want to deposit: \n";
      double am;
      cin >> am;
      balance_cur = balance_cur + am;
      cout << "Your updated balance is: " << balance_cur << "\n";
   }
   void withdraw()
   {
      double wm = 0.0;
      if(balance_cur > 0)
      {
         cout << "Enter the amount you want to withdraw: \n";
         cin >> wm;
         if(wm <= balance_cur)
         {
            balance_cur = balance_cur - wm;
            cout << "Your updated balance is: " << balance_cur << "\n";
         }
         else
```

```
                cout << "Invalid amount/less funds\n";
            }
            else
                cout << "Current balance is Rs. 0";
            balcheck();
        }
        void balcheck()
        {
            if(balance_cur < min_cur)
            {
                cout << "Penalty of less balance: Rs. 250";
                balance_cur = balance_cur - 250;
            }
        }
    };
    class sav_accnt : public account
    {
    public:
        double balance_sav;
        void sets()
        {
            balance_sav = 0.0;
        }
        void deposit()
        {
            cout << "Enter the amount you want to deposit: \n";
            double am;
            cin >> am;
            balance_sav = balance_sav + am;
            cout << "Your updated balance is: " << balance_sav << "\n";
        }
        void withdraw()
        {
            double wm = 0.0;
            if(balance_sav > 0)
            {
                cout << "Enter the amount you want to withdraw: \n";
                cin >> wm;
                if(wm <= balance_sav)
                {
                    balance_sav = balance_sav - wm;
                    cout << "Your updated balance is: " << balance_sav << "\n";
                }
                else
                    cout << "Invalid amount/less funds\n";
            }
            else
                cout << "Current balance is Rs. 0";
        }
        void interest()
        {
            double intr = (4*balance_sav)/100;
            balance_sav = balance_sav + intr;
            cout << "Interest added = " << intr << "\n";
        }
    };
    int main()
    {
        account obj;
        obj.input();
        if(obj.type == 'C')
        {
            cur_accnt ob1;
            ob1.setv();
            ob1.deposit();
            ob1.withdraw();
        }
        else if(obj.type == 'S')
        {
            sav_accnt ob2;
            ob2.sets();
            ob2.deposit();
            ob2.withdraw();
            ob2.interest();
        }
        else
            cout << "Invalid choice of account type\n";
        return 0;
    }
```

Q11)

```
#include <iostream>
#include <string>
#define min_cur 5000
using namespace std;
class account
```

```
{
public:
    string name;
    long accno;
    char type;
    account()
    {
        cout << "Enter your name:\n";
        cin >> name;
        cout << "Enter your account number:\n";
        cin >> accno;
    }
    account(int x)
    {

    }
};
class cur_accnt : public account
{
public:
    double balance_cur;
    cur_accnt()
    {
        balance_cur = 0.0;
    }
    void deposit()
    {
        cout << "Enter the amount you want to deposit: \n";
        double am;
        cin >> am;
        balance_cur = balance_cur + am;
        cout << "Your updated balance is: " << balance_cur << "\n";
    }
    void withdraw()
    {
        double wm = 0.0;
        if(balance_cur > 0)
        {
            cout << "Enter the amount you want to withdraw: \n";
            cin >> wm;
            if(wm <= balance_cur)
            {
                balance_cur = balance_cur - wm;
                cout << "Your updated balance is: " << balance_cur << "\n";
            }
            else
                cout << "Invalid amount/less funds\n";
        }
        else
            cout << "Current balance is Rs. 0";
        balcheck();
    }
    void balcheck()
    {
        if(balance_cur < min_cur)
        {
            cout << "Penalty of less balance: Rs. 250";
            balance_cur = balance_cur - 250;
        }
    }
};
class sav_accnt : public account
{
public:
    double balance_sav;
    sav_accnt()
    {
        balance_sav = 0.0;
    }
    void deposit()
    {
        cout << "Enter the amount you want to deposit: \n";
        double am;
        cin >> am;
        balance_sav = balance_sav + am;
        cout << "Your updated balance is: " << balance_sav << "\n";
    }
    void withdraw()
    {
        double wm = 0.0;
        if(balance_sav > 0)
        {
            cout << "Enter the amount you want to withdraw: \n";
            cin >> wm;
            if(wm <= balance_sav)
            {
```

```
            balance_sav = balance_sav - wm;
            cout << "Your updated balance is: " << balance_sav << "\n";
         }
         else
            cout << "Invalid amount/less funds\n";
      }
      else
         cout << "Current balance is Rs. 0";
   }
   void interest()
   {
      double intr = (4*balance_sav)/100;
      balance_sav = balance_sav + intr;
      cout << "Interest added = " << intr << "\n";
   }
};
int main()
{
   cout << "Enter your account type: 'C' for current, 'S' for savings:\n";
   char t;
   cin >> t;
   account obj(1);
   if(t == 'C')
   {
      cur_accnt ob1;
      ob1.deposit();
      ob1.withdraw();
   }
   else if(t == 'S')
   {
      sav_accnt ob2;
      ob2.deposit();
      ob2.withdraw();
      ob2.interest();
   }
   else
      cout << "Invalid choice of account type\n";
   return 0;
}
```

Q12)

```
#include <iostream>
using namespace std;
```

```
class funovr
{
public:
    void print()
    {
        cout << "Function 1\n";
    }
    void print(int x)
    {
        cout << "Function 2\n";
    }
    void print(char x)
    {
        cout << "Function 3\n";
    }
};
int main()
{
    funovr obj;
    obj.print();
    obj.print(2);
    obj.print('c');
    return 0;
}
```