Assignment 2

COEN 275 Assignment - 2 (10 Marks)

We live in the world of social media. Today, there are around 4.26 billion people on social media which roughly estimate to 58% of the world's population, the number is likely to scale to around 6 billion in next 4 years. Social media have different kinds of users, some users are purely consumers while other are content creators, let's design a simple social media interface by answering the following questions - (10 marks)

Q1. Implement a User class storing a username and number_of_followers (representing number of followers user have) as data members. Add a single constructor accepting two parameters, the username and number_of_followers. Provide appropriate getters (accessors) and setters (mutators). Add a copy constructor, move constructor and a destructor.

Ans:

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

class User
{
private:
  string UserName;
  string number_of_followers;

public:
    User ()
  {
  }
//Parameterized constructor
  User (string UserName, string number_of_followers)
  {
    cout << "Parameterized constructor called\n";
    this->UserName = UserName;
    this->number_of_followers = number_of_followers;
  }
//Copy constructor
```

```
  User (const User & obj)
  {
    cout << "Copy constructor called\n";
    this->UserName = obj.UserName;
    this->number_of_followers = obj.number_of_followers;
  }
//Move constructor
  User (User && obj1)
  {
    cout << "Move constructor called\n";
    UserName = obj1.UserName;
    number_of_followers = obj1.number_of_followers;
    cout << "User: " << obj1.getUserName () << "; No Of Followers: " <<
obj1.getnumber_of_followers () <<"\n";
    obj1.UserName.erase();
    obj1.number_of_followers.erase();
  }
//Destructor
  ~User ()
  {
    cout << "Destructor called!!\n";
  }
//Getters and Setters
  void setUserName (string UserName)
  {
    this->UserName = UserName;
  }
  string getUserName ()
  {
    return UserName;
  }

  string getnumber_of_followers ()
  {
    return number_of_followers;
  }
  void setnumber_of_followers (string number_of_followers)
  {
    this->number_of_followers = number_of_followers;
  }
};

int main ()
{
```

```
  User u1 ("Alex", "100 K");
  vector < User > vec;
  cout <<"User: " << u1.getUserName () << "; No Of Followers: " << u1.getnumber_of_followers ()
<< "\n";
  User u2 = u1;
  cout <<"User: " << u2.getUserName () << "; No Of Followers: " << u2.getnumber_of_followers ()
<< "\n";
  vec.push_back (User{"Ellen", "50 M"});
  return 0;
}
```

Output-

```
Parameterized constructor called
User: Alex; No Of Followers: 100 K
Copy constructor called
User: Alex; No Of Followers: 100 K
Parameterized constructor called
Move constructor called
User: Ellen; No Of Followers: 50 M
Destructor called!!
Destructor called!!
Destructor called!!
Destructor called!!
```

Q2. Add a new data member to store the number_of_following (representing number of users a user follow) of a user and provide a getter & setter. Add a new constructor that accepts three parameters, a username, number_of_followers and number_of_following. Modify the original two parameter constructor to automatically set number_of_following for a given username and number_of_followers and delegate the actual construction work to the new three parameter constructor.

Ans:

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

class User
{
```

```
private:
  string UserName;
  string number_of_followers;
  string number_of_following;

public:
    User ()
  {
  }
//Parameterized constructor
  User (string UserName, string number_of_followers, string number_of_following)
  {
    cout << "Three Parameterized constructor called\n";
    this->UserName = UserName;
    this->number_of_followers = number_of_followers;
    this->number_of_following = number_of_following;
  }
 //Delegation
    User(string UserName, string number_of_followers) :
    User(UserName, number_of_followers, "150")
   {
    cout <<"Two parameterized constructor delegating work to three parameterized is called\n";
   }
//Copy constructor
  User (const User & obj)
  {
    cout << "Copy constructor called\n";
    this->UserName = obj.UserName;
    this->number_of_followers = obj.number_of_followers;
    this->number_of_following = obj.number_of_following;
  }
//Move constructor
  User (User && obj1)
  {
    cout << "Move constructor called\n";
    UserName = obj1.UserName;
    number_of_followers = obj1.number_of_followers;
    number_of_following = obj1.number_of_following;
    cout << "User: " << obj1.getUserName () << "; No Of Followers: " <<
obj1.getnumber_of_followers ()
    << "; No Of Followings: " << obj1.getnumber_of_following () <<"\n";
    obj1.UserName.erase();
    obj1.number_of_followers.erase();
    obj1.number_of_following.erase();
```

```
 }
//Destructor
 ~User ()
 {
  cout << "Destructor called!!\n";
 }

//Getters and Setters
 void setUserName (string UserName)
 {
  this->UserName = UserName;
 }
 string getUserName ()
 {
  return UserName;
 }

 string getnumber_of_followers ()
 {
  return number_of_followers;
 }
 void setnumber_of_followers (string number_of_followers)
 {
  this->number_of_followers = number_of_followers;
 }

 string getnumber_of_following ()
 {
  return number_of_following;
 }
 void setnumber_of_following (string number_of_following)
 {
  this->number_of_following = number_of_following;
 }

};

int main ()
{
 User u1 ("Alex", "100 K");
 cout <<"User: " << u1.getUserName () << "; No Of Followers: " << u1.getnumber_of_followers ()
<< "; No Of Following: "
 << u1.getnumber_of_following ()<< "\n";
 User u2 = u1;
```

```
  cout <<"User: " << u2.getUserName () << "; No Of Followers: " << u2.getnumber_of_followers ()
<< "; No Of Following: "
  << u2.getnumber_of_following ()<< "\n";
  vector < User > vec;
  vec.push_back (User{"Ellen", "50 M"});
  return 0;
}
```

Output-

```
Three Parameterized constructor called
Two parameterized constructor delegating work to three parameterized is called
User: Alex; No Of Followers: 100 K; No Of Following: 150
Copy constructor called
User: Alex; No Of Followers: 100 K; No Of Following: 150
Three Parameterized constructor called
Two parameterized constructor delegating work to three parameterized is called
Move constructor called
User: Ellen; No Of Followers: 50 M; No Of Followings: 150
Destructor called!!
Destructor called!!
Destructor called!!
Destructor called!!
```

Q3. Take the user class from Q1, and add a derived class called Creator. The creator class adds two data members, a creator_id and number_of_reels_created. Provide appropriate constructor. From Creator, derive two more classes called TechCreator and NonTechCreator.

Ans:

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

class User
{
private:
  string UserName;
```

```
  string number_of_followers;

public:
   User ()
  {
  }
//Parameterized constructor
  User (string UserName, string number_of_followers)
  {
   cout << "Parameterized constructor called\n";
   this->UserName = UserName;
   this->number_of_followers = number_of_followers;
  }
//Copy constructor
  User (const User & obj)
  {
   cout << "Copy constructor called\n";
   this->UserName = obj.UserName;
   this->number_of_followers = obj.number_of_followers;
  }

//Move constructor
  User (User && obj1)
  {
   cout << "Move constructor called\n";
   UserName = obj1.UserName;
   number_of_followers = obj1.number_of_followers;
   cout << "User: " << obj1.getUserName () << "; No Of Followers: " <<
obj1.getnumber_of_followers () <<"\n";
   obj1.UserName.erase();
   obj1.number_of_followers.erase();
  }
//Destructor
  ~User ()
  {
   cout << "Destructor called!!\n";
  }
//Getters and Setters
  void setUserName (string UserName)
  {
   this->UserName = UserName;
  }
  string getUserName ()
  {
```

```
  return UserName;
 }
 string getnumber_of_followers ()
 {
  return number_of_followers;
 }
 void setnumber_of_followers (string number_of_followers)
 {
  this->number_of_followers = number_of_followers;
 }
};

class Creater : public User
{
  int creator_id;
  int number_of_reels_created;
  public:
  Creater()
  {
  cout <<"Creater Default Constructor called\n";
  }
  Creater(int creator_id, int number_of_reels_created)
  {
  cout <<"Parameterized Creater Constructor called\n";
  this->creator_id = creator_id;
  this->number_of_reels_created = number_of_reels_created;
  }
//Destructor
  ~Creater()
  {
  cout<<"Creater Destructor called!!\n";
  }
  int getcreator_id(){
  return creator_id;
  }
  void setcreator_id(int creator_id)
  {
  this->creator_id = creator_id;
  }
  int getnumber_of_reels_created(){
  return number_of_reels_created;
  }
  void setnumber_of_reels_created(int number_of_reels_created)
  {
```

```
   this->number_of_reels_created = number_of_reels_created;
   }
};

class TechCreator  : public Creater
{
public:
TechCreator (){
cout<<"TechCreator  Constructor called\n";
}
//Destructor
~TechCreator (){
cout<<"TechCreator  Destructor called!!\n";
}
};

class NonTechCreator : public Creater{
public:
NonTechCreator (){
cout<<"NonTechCreator  Constructor called\n";
}
//Destructor
~NonTechCreator (){
cout<<"NonTechCreator  Destructor called!!\n";
}
};

int main ()
{
  TechCreator TC;
  NonTechCreator NC;
  Creater c(123,10);
  User u1 ("Alex", "100 K");
  vector < User > vec;
  cout <<"User: " << u1.getUserName () << "; No Of Followers: " << u1.getnumber_of_followers ()
  << "; Creater ID: " << c.getcreator_id () << "; No Of Reels Created: " <<
c.getnumber_of_reels_created () << "\n";
  User u2 = u1;
  cout <<"User: " << u2.getUserName () << "; No Of Followers: " << u2.getnumber_of_followers ()
  << "; Creater ID: " << c.getcreator_id () << "; No Of Reels Created: " <<
c.getnumber_of_reels_created () << "\n";
  vec.push_back (User{"Ellen", "50 M"});
  return 0;
}
```

Output –

```
Creater Default Constructor called
TechCreator  Constructor called
Creater Default Constructor called
NonTechCreator  Constructor called
Parameterized Creater Constructor called
Parameterized constructor called
User: Alex; No Of Followers: 100 K; Creater ID: 123; No Of Reels Created: 10
Copy constructor called
User: Alex; No Of Followers: 100 K; Creater ID: 123; No Of Reels Created: 10
Parameterized constructor called
Move constructor called
User: Ellen; No Of Followers: 50 M
Destructor called!!
Destructor called!!
Destructor called!!
Destructor called!!
Creater Destructor called!!
Destructor called!!
NonTechCreator  Destructor called!!
Creater Destructor called!!
Destructor called!!
TechCreator  Destructor called!!
Creater Destructor called!!
Destructor called!!
```

Q4. Continuing with your solution, add introduceMe() method to the User class returning a string representation of a user. Override this method in the Creator, TechCreator and NonTechCreator classes to build up a complete string representation by delegating part of their work to parent classes.

Ans:

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

class User
{
private:
  string UserName;
  string number_of_followers;

public:
   User ()
```

```
  {
  }

//Parameterized constructor
  User (string UserName, string number_of_followers)
  {
    cout << "Three Parameterized constructor called\n";
    this->UserName = UserName;
    this->number_of_followers = number_of_followers;
  }
//Copy constructor
  User (const User & obj)
  {
    cout << "Copy constructor called\n";
    this->UserName = obj.UserName;
    this->number_of_followers = obj.number_of_followers;
  }
//Move constructor
  User (User && obj1)
  {
    cout << "Move constructor called\n";
    UserName = obj1.UserName;
    number_of_followers = obj1.number_of_followers;
    cout << "User: " << obj1.getUserName () << "; No Of Followers: " <<
obj1.getnumber_of_followers () <<"\n";
    obj1.UserName.erase();
    obj1.number_of_followers.erase();
  }
//Destructor
  ~User ()
  {
    cout << "Destructor called!!\n";
  }
//Getters and Setters
  void setUserName (string UserName)
  {
    this->UserName = UserName;
  }
  string getUserName ()
  {
    return UserName;
  }
  string getnumber_of_followers ()
  {
```

```
   return number_of_followers;
  }
  void setnumber_of_followers (string number_of_followers)
  {
   this->number_of_followers = number_of_followers;
  }
  //introduceMe() method
   virtual string introduceMe()
   {
   return "introduceMe()--->"+getUserName()+" "+getnumber_of_followers()+"\n";
   }
};

class Creater : public User
{
   int creator_id;
   int number_of_reels_created;
   public:
   Creater()
   {
   cout <<"Creater Default Constructor called\n";
   }
   Creater(int id, int number_of_reels, string UserName, string
number_of_followers):User(UserName,number_of_followers)
   {
   cout <<"Parameterized Creater Constructor called\n";
   this->creator_id = id;
   this->number_of_reels_created = number_of_reels;
   }
//Destructor
   ~Creater()
   {
   cout<<"Creater Destructor called!!\n";
   }
   int getcreator_id(){
   return creator_id;
   }
   void setcreator_id(int creator_id)
   {
   this->creator_id = creator_id;
   }
   int getnumber_of_reels_created(){
   return number_of_reels_created;
   }
```

```
   void setnumber_of_reels_created(int number_of_reels_created)
   {
   this->number_of_reels_created = number_of_reels_created;
   }
   //introduceMe() method
   virtual string introduceMe()
   {
   return " introduceMe()--->" +to_string(getcreator_id())+","
+to_string(getnumber_of_reels_created())+" "+ User::introduceMe()+"\n";
   }
};

class TechCreator  : public Creater
{
public:
using Creater::Creater;
TechCreator (){
cout<<"TechCreator  Constructor called\n";
}
//Destructor
~TechCreator (){
cout<<"TechCreator  Destructor called!!\n";
}
string introduceMe()
{
return "TechCreater:" + Creater::introduceMe();
}
};

class NonTechCreator : public Creater{
public:
using Creater::Creater;
NonTechCreator (){
cout<<"NonTechCreator  Constructor called\n";
}
//Destructor
~NonTechCreator (){
cout<<"NonTechCreator  Destructor called!!\n";
}
string introduceMe()
{
return "NonTechCreator:" + Creater::introduceMe();
}
};
```

```
int main ()
{
  User u1 ("Alex", "100 K");
  cout<<u1.introduceMe()<<"\n";
  Creater c(123,10,"Alex","100 K");
  cout<<c.introduceMe()<<"\n";
  TechCreator tc(999,999,"Gopi","99 K");
  cout<<tc.introduceMe()<<"\n";
   NonTechCreator nc(420,420,"Mike","899");
  cout<<nc.introduceMe()<<"\n";

  cout <<"User: " << u1.getUserName () << "; No Of Followers: " << u1.getnumber_of_followers ()
  << "; Creater ID: " << c.getcreator_id () << "; No Of Reels Created: " <<
c.getnumber_of_reels_created () << "\n";
  User u2 = u1;
  cout <<"User: " << u2.getUserName () << "; No Of Followers: " << u2.getnumber_of_followers ()
  << "; Creater ID: " << c.getcreator_id () << "; No Of Reels Created: " <<
c.getnumber_of_reels_created () << "\n";

  vector < User > vec;
  vec.push_back (User{"Ellen", "50 M"});
  return 0;
}
```

Output-

```
Three Parameterized constructor called
introduceMe()--->Alex 100 K

Three Parameterized constructor called
Parameterized Creater Constructor called
introduceMe()--->123,10 introduceMe()--->Alex 100 K


Three Parameterized constructor called
Parameterized Creater Constructor called
TechCreater:introduceMe()--->999,999 introduceMe()--->Gopi 99 K


Three Parameterized constructor called
Parameterized Creater Constructor called
NonTechCreator:introduceMe()--->420,420 introduceMe()--->Mike 899


User: Alex; No Of Followers: 100 K; Creater ID: 123; No Of Reels Created: 10
Copy constructor called
User: Alex; No Of Followers: 100 K; Creater ID: 123; No Of Reels Created: 10
Three Parameterized constructor called
Move constructor called
User: Ellen; No Of Followers: 50 M
Destructor called!!
Destructor called!!
Destructor called!!
NonTechCreator  Destructor called!!
Creater Destructor called!!
Destructor called!!
TechCreator  Destructor called!!
Creater Destructor called!!
Destructor called!!
Creater Destructor called!!
Destructor called!!
Destructor called!!
```

Q5. Define a vector to store a mix of Creator, TechCreator and NonTechCreator and fill it with some test data. Finally use a range based for loop to call introduceMe() on all of the elements in the vector.
Submit the code and output (screenshot is a must) files.

Ans:

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

class User
{
private:
  string UserName;
  string number_of_followers;
public:
    User ()
  {
  }
  //Parameterized constructor
  User (string UserName, string number_of_followers)
  {
    cout << "Three Parameterized constructor called\n";
    this->UserName = UserName;
    this->number_of_followers = number_of_followers;
  }
//Copy constructor
  User (const User & obj)
  {
   cout << "Copy constructor called\n";
    this->UserName = obj.UserName;
    this->number_of_followers = obj.number_of_followers;
  }

//Move constructor
  User (User && obj1)
  {
    cout << "Move constructor called\n";
    UserName = obj1.UserName;
    number_of_followers = obj1.number_of_followers;
    cout << "User: " << obj1.getUserName () << "; No Of Followers: " <<
obj1.getnumber_of_followers () <<"\n";
   obj1.UserName.erase();
```

```
   obj1.number_of_followers.erase();
  }
//Destructor
  ~User ()
  {
   cout << "Destructor called!!\n";
  }
//Getters and Setters
  void setUserName (string UserName)
  {
   this->UserName = UserName;
  }
  string getUserName ()
  {
   return UserName;
  }
  string getnumber_of_followers ()
  {
   return number_of_followers;
  }
  void setnumber_of_followers (string number_of_followers)
  {
   this->number_of_followers = number_of_followers;
  }
  //introduceMe() method
    virtual string introduceMe()
    {
    return "introduceMe()--->"+getUserName()+" "+getnumber_of_followers()+"\n";
    }
};

class Creater : public User
{
    int creator_id;
    int number_of_reels_created;
    public:
    Creater()
    {
    cout <<"Creater Default Constructor called\n";
    }
    Creater(int id, int number_of_reels, string UserName, string
number_of_followers):User(UserName,number_of_followers)
    {
    cout <<"Parameterized Creater Constructor called\n";
```

```
   this->creator_id = id;
   this->number_of_reels_created = number_of_reels;
   }
//Destructor
   ~Creater()
   {
   cout<<"Creater Destructor called!!\n";
   }
   int getcreator_id(){
   return creator_id;
   }
   void setcreator_id(int creator_id)
   {
   this->creator_id = creator_id;
   }
   int getnumber_of_reels_created(){
   return number_of_reels_created;
   }
   void setnumber_of_reels_created(int number_of_reels_created)
   {
   this->number_of_reels_created = number_of_reels_created;
   }
   //introduceMe() method
   virtual string introduceMe()
   {
   return " introduceMe()--->" +to_string(getcreator_id())+
to_string(getnumber_of_reels_created())+ User::introduceMe()+"\n";
   }
};

class TechCreator  : public Creater
{
public:
using Creater::Creater;
TechCreator (){
cout<<"TechCreator  Constructor called\n";
}
//Destructor
~TechCreator (){
cout<<"TechCreator  Destructor called!!\n";
}
string introduceMe()
{
return "TechCreater:" + Creater::introduceMe();
```

```
}
};

class NonTechCreator : public Creater{
public:
using Creater::Creater;
NonTechCreator (){
cout<<"NonTechCreator  Constructor called\n";
}
//Destructor
~NonTechCreator (){
cout<<"NonTechCreator  Destructor called!!\n";
}
string introduceMe()
{
return "NonTechCreator:" + Creater::introduceMe();
}
};

int main ()
{
  vector<User>U;
  U.push_back (Creater(123,10,"Alex","100 K"));
  U.push_back (TechCreator(444,3243,"Fox","88 K"));
  U.push_back (NonTechCreator(120,50,"John","30 K"));
  for (auto temp : U)
       {
           cout << temp.introduceMe() <<"\n";
           }

  /*User u1 ("Merry", "430 K");
  Creater c(45,25,"Merry","430 K");
  cout <<"User: " << u1.getUserName () << "; No Of Followers: " << u1.getnumber_of_followers ()
  << "; Creater ID: " << c.getcreator_id () << "; No Of Reels Created: " <<
c.getnumber_of_reels_created () << "\n";
  User u2 = u1;
  cout <<"User: " << u2.getUserName () << "; No Of Followers: " << u2.getnumber_of_followers ()
  << "; Creater ID: " << c.getcreator_id () << "; No Of Reels Created: " <<
c.getnumber_of_reels_created () << "\n";

  vector < User > vec;
  vec.push_back (User{"Ellen", "50 M"});*/
  return 0;
}
```

Output –

```
User: Alex; No Of Followers: 100 K
Creater Destructor called!!
Destructor called!!
Three Parameterized constructor called
Parameterized Creater Constructor called
Move constructor called
User: Fox; No Of Followers: 88 K
Copy constructor called
Destructor called!!
TechCreator  Destructor called!!
Creater Destructor called!!
Destructor called!!
Three Parameterized constructor called
Parameterized Creater Constructor called
Move constructor called
User: John; No Of Followers: 30 K
Copy constructor called
Copy constructor called
Destructor called!!
Destructor called!!
NonTechCreator  Destructor called!!
Creater Destructor called!!
Destructor called!!
Copy constructor called
introduceMe()--->Alex 100 K

Destructor called!!
Copy constructor called
introduceMe()--->Fox 88 K

Destructor called!!
Copy constructor called
introduceMe()--->John 30 K

Destructor called!!
Destructor called!!
Destructor called!!
Destructor called!!
```