BLG252E – Object Oriented Programming
Midterm Exam-2
Answers

QUESTION 1) [15 points]

```
50
60
50
-----
30
15
20
225
35
20
-----
20
```

QUESTION 2a) [30 points]

```
class Account {
  string name;
  float value;
public:
  Account(string, float);
  void deposit(float);
  bool withdraw(float);
  bool withdraw(float, float);
  bool transfer(Account &, float);
  void print();
};
```

```
Account::Account(string cust_name, float init_val=0)
{
  name = cust_name;
  value = init_val;
}
```

```
void Account::deposit(float amount)
{
  value += amount;
}
```

```
bool Account::withdraw(float amount)
{
  if (amount > value)
     return false;
  value -= amount;
  return true;
}
```

```
bool Account::withdraw(float amount, float feeRate)
{
  return withdraw(amount*(1 + feeRate) );
}
```

```
bool Account::transfer(Account &target, float amount)
{
  if (&target == this)
  {
     cout << "Attention : Self transfer not allowed.\n";
     return false;
  }

  if (withdraw(amount, 0.01))
  {
     target.deposit(amount);
     return true;
  }
  else return false;
}
```

```
void Account::print() {
  cout << name << " " << value << endl;
}
```

QUESTION 2b) [10 points]

```
int main()
{
  Account M1("AAA", 1000);
  Account M2("BBB");

  M1.withdraw(500, 0.02);
  M1.deposit(100);
  M1.transfer(M2, 300);
  M2.withdraw(200);
  M2.deposit(400);
  M2.transfer(M1, 100);

  M1.transfer(M1, 70);

  M1.print();
  M2.print();

  return 0;
}
```

QUESTION 3a) [30 points]

```
#define PI 3.14

class GenericShape  // Abstract base class
{
  protected :
  float density;

  public :
  // Constructor
  GenericShape(float density) : density(density) {}

  virtual float volume() =0;  //Pure virtual
  float weight() { return density * volume();}
};
```

```
class Prism : public GenericShape
{
  int a,b,c;

  public :
  // Constructor
  Prism(float density,int a,int b,int c) :
           GenericShape(density), a(a),b(b),c(c) {}

  float volume() {return a*b*c; }
};
```

```
class Sphere : public GenericShape
{
  int r;

  public :
  // Constructor
  Sphere(float density,int r) : GenericShape(density), r(r) {}

  float volume() {return 4*PI*r*r*r/3; }
};
```

```
class Cylinder : public GenericShape
{
  protected:
  int r,h;

  public :
  // Constructor
  Cylinder(float density,int r,int h) :
                      GenericShape(density), r(r),h(h) {}
  float volume() {return PI*r*r*h; }
};
```

```
class Cone : public Cylinder
{
  public :
  // Constructor
  Cone(float density,int r,int h) :
           Cylinder(density,r,h) {}

  float volume() {return PI*r*r*h/3; }
};
```

QUESTION 3b) [15 points]

```
int main()
{
// Declare array of pointers:
GenericShape * pa[10];

pa[0] = new Prism(2,6,8,5);
pa[1] = new Prism(3,5,4,9);
pa[2] = new Sphere(1,3);
pa[3] = new Cylinder(1,4,8);
pa[4] = new Cylinder(4,2,9);
pa[5] = new Cone(3,5,7);
pa[6] = new Prism(1,7,2,1);
pa[7] = new Sphere(2,9);
pa[8] = new Cylinder(1,4,6);
pa[9] = new Cone(1,5,3);

// Calculate and display weights
// of all elements:

for (int i=0; i<10;i++)
   cout << pa[i]->weight() << endl;

 return 0;
}
```