CS1004- Object Oriented Programming
Final Exam Solution

Solution # 01:

```
#include <iostream>
#include <fstream>
#include <conio.h>
using namespace std;
class Student {
        int semestert_hours;
        char name[25];
        float Tqp;
        float cgpa;
        bool warning;
        void getdata(){
                cout<<"\nEnter Name : ";
                cin>>name;
                cout<<"\nEnter semestert_hours : ";
                cin>>semestert_hours;
                cout<<"\n\nEnter Tqp : ";
                cin>>Tqp;
                cgpa=Tqp/semestert_hours;
                if((cgpa<1.5 && semestert_hours<30)|| (cgpa<1.75 && semestert_hours<60) ||(cgpa==2)){
                        warning=true;
                }
                else{
                        warning=false;
                }
        }
        void fetchdata() {
                cout<<"\n\t"<<name<<"\t"<<semestert_hours<<"\t"<<Tqp<<"\t"<<cgpa<<"\t"<<warning;
                }
        public:
        void AddRecord(){
                fstream f;
                Student Stu;
                f.open("Student.txt",ios::app);
                Stu.getdata();
                f.write( (char *) &Stu,sizeof(Stu) );
                f.close();
        }
        void Display(){
                fstream f;
                Student Stu;
                f.open("Student.txt",ios::in|ios::binary);
                cout<<"\n\tName\tsemestert_hours\tTqp\tCGPA\tWarning\n";
                while((f.read((char*)&Stu,sizeof(Stu))) != NULL )
                        Stu.fetchdata();
                        f.close();
        }};
int main(){
 Student S;
```

```
    char ch='n';
    do{
            S.AddRecord();
            cout<<"\n\nDo you want to add another data (y/n) : ";
            ch = getche();
    } while(ch=='y' || ch=='Y');
            cout<<"\nData written  successfully...";
    S.Display();
    //        S.cal_cgpa();
    }
```

Solution # 02:

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;
class Book{
          int BookID;
          string Book_Title;
    float price;
    string Author;
    string Status;
    bool availability;
    public:
      Book(){
                  }
      ~Book(){}
      void setBookID(int BID){  BookID= BID;}
      void setBName(string BTitle){ Book_Title = BTitle;}
      void setprice(float Bprice) { price = Bprice; }
      void setBAuthor(string BAuthor) { Author = BAuthor; }
      void setBStatus(string BStatus) { Status = BStatus; }
      void setBSava(bool Bavailability) { availability = Bavailability; }
      void printBookInfo(){
          cout << "Book ID      : " << BookID << endl;
          cout << "Book Title      : " << Book_Title << endl;
                  cout << "Book Price      : " << price << endl;
                  cout << "Book Author     : " << Author << endl;
                  cout << "Book Status     : " << Status << endl ;
                  cout << "Book Availability     : " << availability << endl <<endl;
                  };

};
int main(){
  // Create a vector of Friend objects
  vector<Book> list;
  int BookID;
        string Book_Title;
  float price;
  string Author;
  string Status;
  bool availability;
Book *f1;
```

```
for (int n=0; n<3; n++)
{
                        cout << "Book ID      : " << endl;
                        cin>>BookID;
        cout << "Book Title      : "  << endl;
        //getline(cin, Book_Title);
        cin>>Book_Title;
                cout << "Book Price      : "  << endl;
                cin>>price;
                cout << "Book Author     : " << endl;
                //getline(cin, Author);
                cin>>Author;
                cout << "Book Status     : " <<  endl ;
                getline(cin, Status);
                cout << "Book Availability     : "   <<endl;
                cin>>availability;
    cin.get(); //clear buffer

    f1 = new Book;
    f1->setBookID(BookID);
    f1->setBName(Book_Title);
    f1->setprice(price);
    f1->setBAuthor(Author);
    f1->setBStatus(Status);
    f1->setBSava(availability);
    list.push_back(*f1);

  }
//  // Now setup an iterator loop through the vector
  vector<Book>::iterator it;
  for ( it = list.begin(); it != list.end(); ++it ) {
    // For each friend, print out their info
    it->printBookInfo();
  }
  return 0;
}
```

Solution # 03:

```
#include<iostream>
using namespace std;
class File{
        int size;
        string location;
        Date created_date;
        Date modified_date;
        public:
                void Open(){
                        cout<<"Open File"<<endl;
                }
                virtual void Print(){
                        cout<<"Print File"<<endl;
                }
};
```

```
class PDF:public File{
        public:
                void Print(){
                        cout<<"Print PDF "<<endl;
                }
};
class ASCII: public File{
        public:
                void Print(){
                        cout<<"Print ASCII "<<endl;
                }
};
class PS:public File{
        public:
                void Print(){
                        cout<<"Print PS "<<endl;
                }
};
int main(){
        File *f1=new PDF();
        File *f2=new ASCII();
        File *f3=new PS();
        f1->Print();
        f2->Print();
        f3->Print();
}
```

Solution # 04:

```
#include<iostream>
using namespace std;
class rationalnumber{
        private:
                int n,d,m;
        public:
        rationalnumber(){
                }
        rationalnumber(int a, int b){
                try{
                        n=a,d=b;
                        if(d<=0){
                                throw d;
                        }
                }
                catch(int){
                        cout<<"incorrect denominator."<<endl;
                        exit(0);
                }
                if(n>d){
                        m=n;
                }
                else{m=d;}
                for(int i=2;i<=m;i++)
                {
```

```
        cout<<result<<endl;
        r1==r2;
}
```

Solution # 05:

Part1:

```
#include<iostream>
#include <string>

using namespace std;
class Time{
        string time;
        int hour;
  int minute;
  int second;
  int total_seconds;
  public:
        void get_time(){
                try{cout<<"Enter time "<<endl;cin>>time;
                        if(time.length()!=8)
                        throw time;
                }
                catch(string invalidTime){cout <<"The value that you entered is invalid, try again.";
                        }
                }
                void convert(){
                        try{
                        hour=time.substr(0,2);
                        if(hour>=0 && hour<=24) throw hour;
                        minute=time.substr(3,5);
                        if(minute>=0 && minute<=60) throw minute;
                        seconds=time.substr(6,8);
                        if(second>=0 && second<=60) throw second;
                        }
                        catch(int invalidH){
                                cout <<"The hours value that you entered is invalid, try again.";
                        }
                        catch(int invalidM){
                                cout <<"The minutes value that you entered is invalid, try again.";
                        }
                        catch(int invalidS){
                                cout <<"The seconds value that you entered is invalid, try again.";
                        }
                }
                void convertIntoSeconds(){
                        total_seconds = (hour*60*60)+(minute*60)+second;
                }
                void Display(){
                        cout<<"Time\t"<<time<<endl;
                        cout<<"Hours\t"<<hour<<endl;
                        cout<<"Minutes\t"<<minute<<endl;
                        cout<<"Seconds\t"<<second<<endl;
                        cout<<"Total Seconds\t"<<total_seconds<<endl;
```