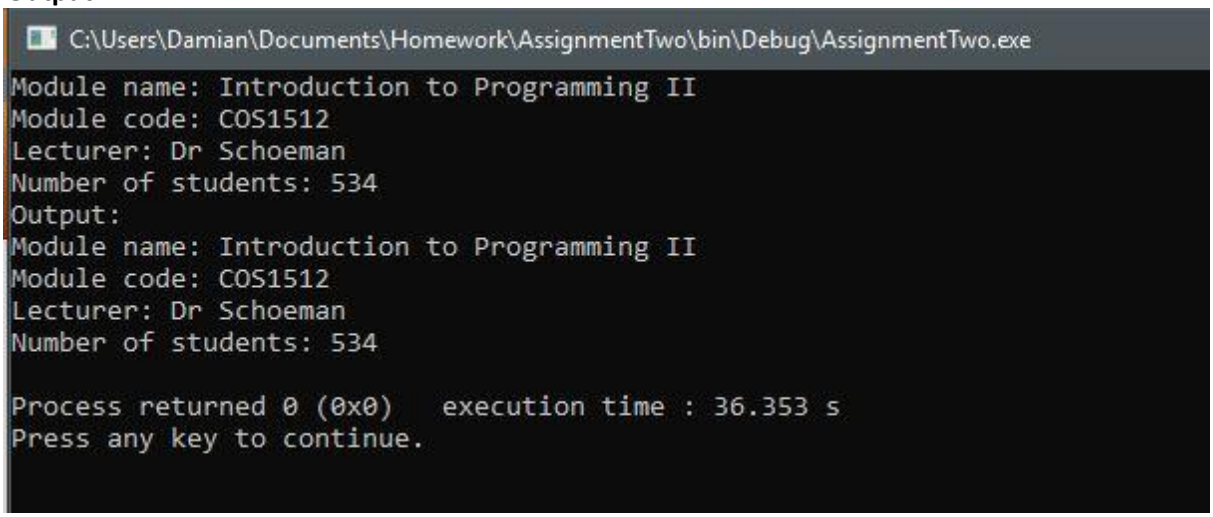
Question 1:

Code:

```
#include<iostream>

using namespace std;

class Module {
   private:
      string moduleName;
      string moduleCode;
      string lecturer;
      int nrStudents;

   public:
      Module(){
         moduleName="";
         moduleCode="0000000";
         lecturer="";
         nrStudents=0;
      }

      Module(string modName, string modCode, string lect, int modnrStudents){
         moduleName=modName;
         moduleCode=modCode;
         lecturer=lect;
         nrStudents=modnrStudents;
      }

      void setModuleName(string modName);
      void setModuleCode(string modCode);
      void setLecturer(string lect);
      void setnrStudents(int modnrStudents);

      string getModuleName();
      string getModuleCode();
      string getLecturer();
      int getnrStudents();
};

void Module::setModuleName(string moduleName){
   moduleName = moduleName;
}

void Module::setModuleCode(string modCode){
   moduleCode = modCode;
}
```

```
void Module::setLecturer(string lect){
   lecturer = lect;
}

void Module::setnrStudents(int modnrStudents){
   nrStudents = modnrStudents;
}

string Module::getModuleName(){
   return moduleName;
}

string Module::getModuleCode(){
   return moduleCode;
}

string Module::getLecturer(){
   return lecturer;
}

int Module::getnrStudents(){
   return nrStudents;
}

int main(){
   string moduleName;
   string moduleCode;
   string lecturer;
   int nrStudents;

   cout<<"Module name: ";
   getline(cin, moduleName);
   cout<<"Module code: ";
   getline(cin, moduleCode);
   cout<<"Lecturer: ";
   getline(cin, lecturer);
   cout<<"Number of students: ";
   cin>>nrStudents;

   Module module = Module(moduleName, moduleCode, lecturer, nrStudents);

   cout<<"Output: "<<endl;
   cout<<"Module name: "<<module.getModuleName()<<endl;
   cout<<"Module code: "<<module.getModuleCode()<<endl;
   cout<<"Lecturer: "<<module.getLecturer()<<endl;
   cout<<"Number of students: "<<module.getnrStudents()<<endl;

   return 0;
}
```

Output:



Question 2

a) They change the scope of variables or methods.

b) A class is a template for objects; a class controls the objects behaviour and attributes.

c) To create an instance of an object with specific attributes that live in memory.

d) To initialise default attributes for an object.

e) A default constructor initialises default attributes for an object, where as an overloaded constructor will initialise the object with custom attributes.

f) To delete an object.

g) To access the attributes of an object.

h) To change or set the attributes of an object

i) To have access to hidden variables and use variables with the same name in different scopes.

j) The scope resolution operator is used to have access to hidden variables and use variables with the same name in different scopes, whereas the dot operator gives access to attributes of a specific object.

k) A member function needs to be called with an instance of its class or structure whereas an ordinary function called be called directly in the main or other functions.

l) Is a class with a defined set of operations and values. It hides operations from the user. The user will now what is the values and operations but not how it does them.

m) You defined its attributes and operations.

n) Re-usable code to implement different types of the abstract data type and having set predefined operations on them.

o) A class that is compiled separately before the rest of the project is compiled or just before its run.

p) Classes can be reused by different classes and only need to be compiled once.

q) A class that is created based on an existing class.

r) It allows a derived class to inherit properties from a base class.

Question 3

- Move the 'return 0;' statement after the 'cout' statement.
- 'double nScore = sodoku.Champion;' will throw an error; Use double 'nScore = sodoku.Score;'

Question 4

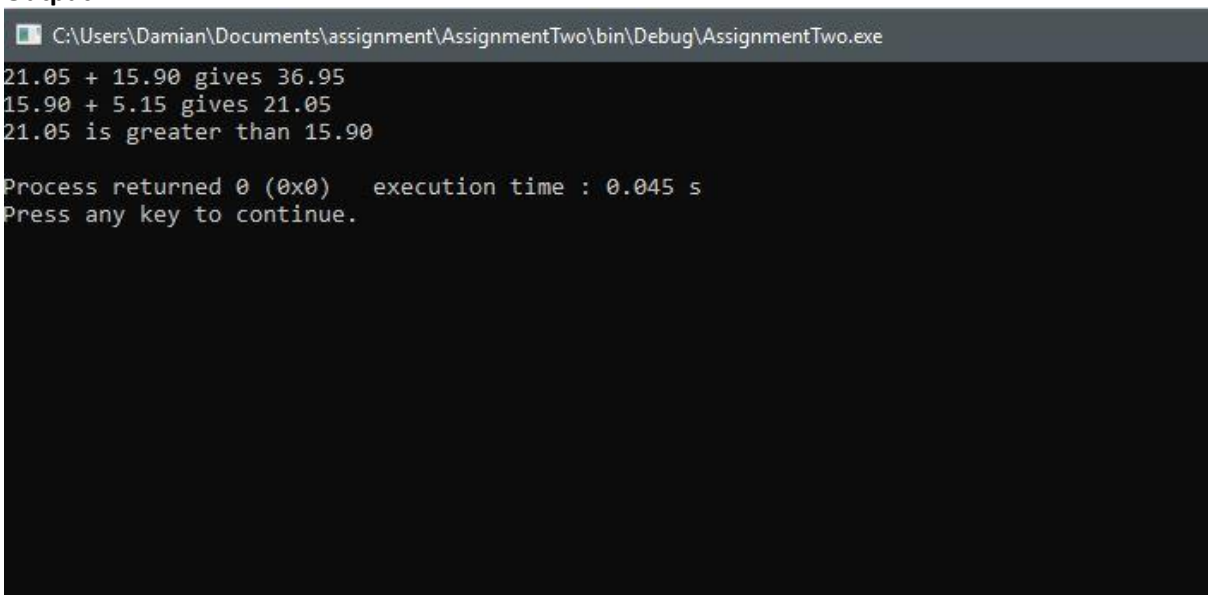
a) Public will make variables and methods available to be modified from outside the class where private only allows access within the class.
b) To initialise variables and methods

```
#include<iostream>
using namespace std;
class Money{
    public:
        Money(){
            rands=0;
            cents=0;
        }; // default constructor
        Money(int r, int c){
            rands=r;
            cents=c;
        }; // constructor
        ~Money(); // destructor

        int theRands() const;
        int theCents() const;

        Money Plus(Money m);
        Money operator+ (Money &m);
//        bool operator> (Money &m);
        bool operator> (Money m);
        bool GreaterThan(Money m);
        friend ostream& operator<<(ostream& os, const Money& m);
    private:
        int rands;
        int cents;
};

int Money::theRands() const { return rands;}
int Money::theCents() const { return cents;}

Money Money::Plus(Money m) {
    std::div_t dv{};
    dv = std::div(cents+m.theCents(),100);
    return Money(rands+m.theRands()+dv.quot,dv.rem);
}

Money Money::operator+(Money &m) {
    int *rand;
```

```
    rand=&m.rands;
    int *cent;
    cent=&m.cents;
    std::div_t dv{};
    dv = std::div(cents+*cent,100);
    return Money(rands+*rand+dv.quot,dv.rem);
}

ostream& operator<<(ostream& os, const Money& m){
    if (m.theCents()>9)
    os<<m.theRands()<<"."<<m.theCents();
    else
        os<<m.theRands()<<".0"<<m.theCents();
}

bool Money::GreaterThan(Money m) {
    if(rands>m.theRands()){
        return true;
    } else if (m.theRands()==rands && m.theCents()>cents){
        return true;
    }
    return false;
}
/*
bool Money::operator>(Money &m){
    int *rand;
    rand=&m.rands;
    int *cent;
    cent=&m.cents;
    cout<<rands<<"b"<<*rand<<endl;
    if(rands>*rand){
        return true;
    } else if (*rand==rands && *cent>cents){
        return true;
    }
    return false;
}*/

bool Money::operator>(Money m) {
    if(rands>m.theRands()){
        return true;
    } else if (m.rands==rands && m.cents>cents){
        return true;
    }
}

Money::~Money() {
}

int main()
{
    Money m1=Money();
    Money m2=Money(15,90);
    Money m3=Money(5,15);
    m1 = m2.Plus(m3);
    cout<<m1<<" + "<<m2<<" gives "<<m1.Plus(m2)<<endl;
    m1 = m2 + m3;
    cout << m2 << " + " << m3 << " gives " << m1 << endl;
    if (m1>m2)
    cout << m1 << " is greater than " << m2 << endl;
    else {
```

```
        cout << m2 << " is less than " << m1 << endl;
    }
    m1.~Money();
    m2.~Money();
    m3.~Money();
        Return 0
}
```

Output:



Question 5:

Code:

Cpp file:

```
#include <iostream>
#include <fstream>
#include <fcntl.h>

using namespace std;

class MedicalAidMembers{
private:
    string name;
    int number;
    int nrDependants;
    string dependants[6];
    double contribution;
public:
    void setContribution(double contribution);

    string getName() ;
    int getNumber() ;

    MedicalAidMembers();
    MedicalAidMembers(string name, int number);
```

```
    ~MedicalAidMembers();
    void addDependant(const string dependant);
    friend ostream& operator<<(ostream& io, MedicalAidMembers
medicalAidMember);
};

MedicalAidMembers::MedicalAidMembers() :
        name(""), number(0), dependants({"","","","","",""}),
contribution(0), nrDependants(0) {}

MedicalAidMembers::MedicalAidMembers(string name, int number) : name(name),
number(number), nrDependants(0),

dependants({"","","","","",""}), contribution(1000) {}

string MedicalAidMembers::getName() {
    return name;
}

int MedicalAidMembers::getNumber() {
    return number;
}

void MedicalAidMembers::addDependant(const string dependant) {
    if(nrDependants<=6){
        nrDependants+=1;
        dependants->append(dependant);
    }
    else cout<<"Member already has 6 dependants"<<endl;
}

ostream &operator<<(ostream &io, MedicalAidMembers medicalAidMember) {
    io<<medicalAidMember.getName()<<endl;
    io<<medicalAidMember.getNumber()<<endl;
    io<<medicalAidMember.nrDependants<<endl;
    for(int i=0;i<medicalAidMember.dependants->length();i++)
        io<<medicalAidMember.dependants<<endl;
    io<<"R "<<medicalAidMember.contribution<<endl;
}

void MedicalAidMembers::setContribution(double contribution) {
    MedicalAidMembers::contribution = contribution;
}
```

H file:

```
#ifndef INTERFACE_H
#define INTERFACE_H

#include <string>
#include <iostream>
#include <fstream>

using namespace std;

class MedicalAidMembers{
public:
    MedicalAidMembers();
    MedicalAidMembers(string , int);
    void setContribution(double );

    string getName() ;
```

```
    int getNumber() ;

    ~MedicalAidMembers();
    void addDependant(const string);
    friend ostream& operator<<(ostream&, MedicalAidMembers);

};

#endif // INTERFACE_H
```

Main.cpp:

```
#include<iostream>
#include "62951718-questionFive.h"
#include <fstream>
#include <stdlib.h>
#include <string>

using namespace std;

int main(){
    ofstream medicalAidFile;
    medicalAidFile.open("MedicalAidData.dat",ios::out);
    ifstream staff;
    staff.open("staff.dat", ios::in);
    string name1;
    int number1=0;
    int nrDependants1=0;
    MedicalAidMembers member;
    string dependant1;
    staff.get()>>name1;
    cout<<name1;
    while (name1.length()){
        number1=rand();
        cout<<name1<<"'s medical aid number is "<<number1<<endl;
        cout<<"How many dependants does this member have? ";
        cin>>nrDependants1;

        member=MedicalAidMembers(name1,number1);

        for(int i=0;i<nrDependants1;i++){
            cout<<"Enter name of dependant: ";
            getline(cin,dependant1);
            member.addDependant(dependant1);
        }
        member.setContribution(nrDependants1*150.55);
        medicalAidFile<<member;
    }
    member.~MedicalAidMembers();
    return 0;
}
```

Question 6