Name : Shubhansu Kr. Singh Section : K21GP
Reg No : 12104991 Roll No : RK21GPA33

Q-1 File Handling

Ans-

Algorithm :

(i) Take input n (no. of books).
(ii) Create n objects of class book using array.
(iii) Set info of the n objects (input from user).
(iv) Write the data in file (record).
(v) Take input from user (book id) to update.
(vi) Read from file
↳ if book id exists → update value and display the new value on screen as well as write in file.
↳ else show invalid book id.

Highlights :

(i) writeinfile function : Takes an array of object & write the data in file.
(ii) readfromfile function : Reads the data from file and displays on screen.
(iii) update_infile function : Takes the array of object & book id to update as argument & update that data in file.

Name: Shubhansu Kr. Singh Section: K21GP
Reg No: 12104991 Roll No: RK21GPA33

(Q-1) Code:

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

class Book
{
  public:
    int book_id;
    string book_name, book_author;
    float book_price;

    void setInfo() {
        cout <<"Enter book id, name, author,
                                price" <<endl;
        cin >> book_id;
        cin.ignore;
        getline (cin, book_name);
        getline (cin, book_author);
        cin >> book_price;
    }
  void getInfo() {

      cout << book_id << ", " << book_name <<
          ", " << book_author << ", " <<
                  book_price << endl;
    }
};
```

Name : Shubhanshu kr singh Section : K21GP
Reg No : 12104991 Roll No : RK21GPA33.

```
void writeInFile (Book *obj, int n) {
{
    fstream fout ("record.txt", ios::out);
    for (int i=0; i<n; i++)
    {
        if (obj[i].book_price < 500)
        {
            fout.write ((char*) &obj[i], sizeof (obj[i]));
        }
    }
}

void readFromFile ()
{
    fstream file ("record.txt", ios::in);
    string line;
    while (file)
    {
        getline (file, line);
        cout << line << endl;
    }
}
```

Name: Shubhansu Kr. Singh Section: K21GP
Reg No: 12104991 Roll No: RK21GPA33.

```
void updateFile (Book *obj, int n, int temp_id)
{
    for (int i=0; i<n; i++)
    {
        if (obj[i].book_id == temp_id)
        {
            cout << "Enter updated data" << endl;
            obj[i].setInfo();
        }
    }
    writeInFile (obj, n);
}

int main () {
{
    int n;
    cout << "Enter the number of books : ";
    cin >> n;

    Book obj[n];
    for (int i=0; i<n; i++)
    {
        obj[i].setInfo();
    }
    writeInFile (obj, n);
```

```
int temp_id;
cout << "Enter book_id to update";
cin >> temp_id;

fstream fin;
fin.open("record.txt", ios::in);

int flag = 0;
Book temp;

while (!fin.eof())
{
    fin.read((char*)&temp, sizeof(temp));
    if (temp_id == temp.bookid)
    {
        flag = 1;
        updateFile(obj, temp_id);
    }
}
if (flag) {
    readFromFile();
} else {
    cout << "Record not found" << endl;
}
return 0;
```

3.

Name : Shubhansu Kr Singh Section : K21GP
Reg No : 12104991 Roll No : RK21GPA33

Q-2 WAP to overload operator in the same program by writing suitable operator friend function for the following expression:

$$07 = ((01 >= 02) + (03 \&\& 04) * (05 < 06)).$$

Ans

greater than or equal to (>=), less than (<) & logical and (&&) operator evaluates the operands & return the result in true or false (0 or 1).

These operators have two operands, so we need to pass two object as argument/parameter to the friend function. However the assignment cannot be overloaded using friend function, therefore overloading assignment operator using member function.

code :

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

class Overloading
{
    int data;
  public:
```

Name: Shubhansu Kr Singh
Roll No: 12104991
Section: K21GP
Roll No: RK21GPA33

```
    Overloading () { }
    Overloading (int data) : data (data) { }
    friend bool operator >= (Overloading &obj1,
                             Overloading &obj2);
    friend bool operator < (Overloading &obj1,
                            Overloading &obj2);
    friend bool operator && (Overloading &obj1,
                             Overloading &obj2);
    void operator = (int &x);
    int getData () {
        return data;    // Access private data
    }
};
//
bool operator >= (Overloading &obj1, Overloading &obj2)
{
    if (obj1.data >= obj2.data) {
        return true;    // return true if
    }                   // obj1.data is greater
    return false;       // or equal to obj2.data
}

bool operator < (Overloading &obj1, Overloading &obj2)
{
    if (obj1.data < obj2.data) {
        return true;
```

```
        }
        return false;
}

bool operator && (Overloading &obj1, Overloading &obj2)
{
        if (obj1.data && obj2.data)
        {
                return true;
        }
        return false;
}

void Overloading :: operator = (int &x);
{
    data = x;
}

int main () {
        Overloading O1(1), O2(2), O3(0), O4(2), O5(2),
                                        O6(8), O7;
        O7 = ((O1 >= O2) + (O3 && O4)*(O5 < O6));
        cout << O7.getData() << endl;
        return 0;
}.
```

Name : Shubhansu kr Singh Section : K21GP
Reg No : 12104991 Roll No : RK21GPA33

Q-3 Write a program to perform class to basic conversion.

Ans- The conversion from class to basic data type is achieved using cast overloading.

syntax :

```
operator data-type ()
{
    // code (conversion member function)
    // return <val>          of class.
}
```

Code : Class to Basic data type :

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

class Triangle                    // class triangle
{
    float base, height
public:
    Triangle () {} ; // Using constructor to
    Triangle (float b, float h) { // initialise data
        base = b;                 // member.
        height = h;
    }
```

Name: Shubhanan Kr Singh Section: R21GP
Reg No: 12104991 Roll No: RK21GPA33

```
// cast overloading for float.
    operator float () {
        int area = 0.5 * base * height;
        return area;
    }
};

int main () {
    Triangle t1(4, 8);      // constructor call
    float area = t1;        // conversion from
    cout << area << endl;   // class to float.
    return 0;
}
```

Name : Shubhansu kr. Singh Section : K21GP
Reg No. : 12104991 Roll No : RK21GPA33

Q-4 Discuss various application of Initializer list with the help of suitable examples.

Sol^n Initializer list is used to initialise the data members of a class. The list of members to be initialised is indicated by constructor as a comma seprated list followed by a colon. Initializer list can be used with member function as well.

eg - <class_name>(para1, para2, ...) : m1(para1), m2(para2), ... { };

(<class_name> : constructor; para1, para2 : parameters; m1, m2 : data member)

Applications of Initializer list :

(i) when parameter name is same as that of data member :

```
class A {
    int i;
  public:
    A(int i) {
      this.i = i ; };
};
```

We need to use this pointer to refer to the data member.

```
class A {
    int i;
  public:
    A(int i) : i(i) { };
};
```

(i : data member; i : parameter)

Name : Shubhansu Kr Singh Section : K21GP
Reg No : 12104991 Roll No : RK21GPA33

Q4→ When the parameter name is same as data member, ~~use~~ (as shown in previous example), we need to use this pointer to refer to the data member, otherwise (i = i) wont work, since i is the local variable which is the parameter.

To avoid the use of this pointer & confusion caused by the same name, we can simply use initializer list to assign values.

(ii) For initialization of non-static const data members : Constant data members must be initialized using Initializer list.

eg -
```
class A {
    const int a;
public:
    A (int a) : a(a) { };
};
```

Reason for initializing the constant data member in the initializer list is because no ~~memory~~ memory is allocated seprately for const. data members, it is folded in the symbol table due to which we need to initialize it in the initializer list.

Name : Shubhanu Kr Singh Section : K21GP
Reg No : 12104991 Roll No : RK21GPA33.

(iii) Initialization of reference data members :
Reference members must be initialized using initializer list.

eg-
```
class A {
    int &a;
public :
    // A() {};   --> error.
    // Use initializer list
    A(int &a) : a(a) {};
};
```

(iv) Performance : Initialising data members using initializer list takes less resource than using assignment operator to initialise each & every data member.