OBJECT ORIENTED PROGRAMMING

LAB # 14

ASSIGNMENT

NAME: S.M.IRTIZA
ROLL NO. : 22K-4638
CLASS: 2-F

QUESTION#01:

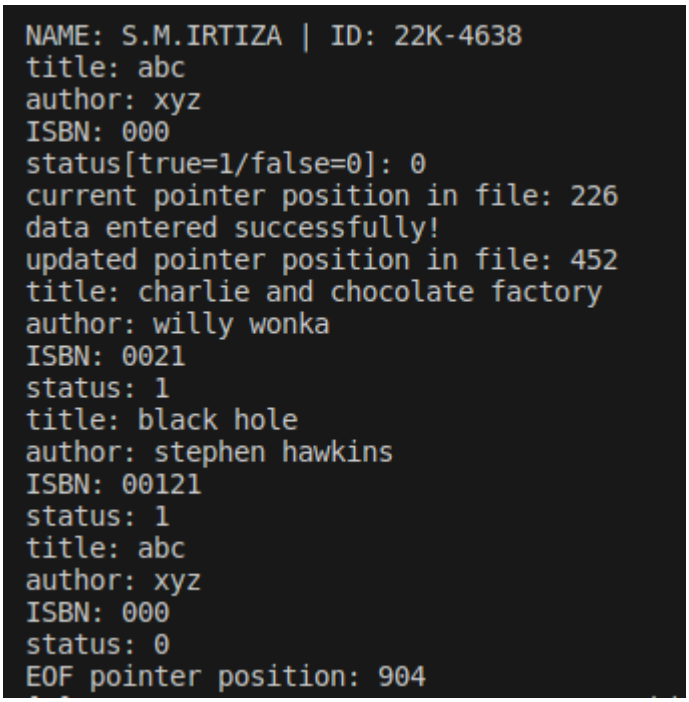
Suppose you are working on a library management system that needs to store information about books, such as their title, author, ISBN, and availability status, in a file. You need to implement a data structure to store this information and provide functionality to read and write the data to a file. You decide to use a binary file to store the book information because it allows for faster read and write operations than a text file. You also want to use file modes to control the behavior of the file stream, seek and tell functions to manipulate the file pointer, and formatted and unformatted input-output operations to interact with the file.

CODE:

```
//"NAME: S.M.IRTIZA | ID: 22K-4638
#include <iostream>
#include <fstream>
#include <cstring>

using namespace std;

class Book {
public:
char title[100];
char author[100];
char ISBN[25];
bool isavailable;
void Input(){
int x;
cout<<"title: ";
fflush(stdin);
cin.getline(title,100);
cout<<"author: ";
fflush(stdin);
cin.getline(author,100);
cout<<"ISBN: ";
fflush(stdin);
cin.getline(ISBN,25);
cout<<"status[true=1/false=0]: ";
cin>>x;
isavailable = (x == 1);
}
};
void AddToFile(Book obj){
fstream write;
write.open("books.dat",ios :: binary | ios :: app);
try{
if(!write){
throw 1;
}
write.write(reinterpret_cast<char*>(&obj),sizeof(obj));
write.close();
cout<<"data entered successfully!"<<endl;
}
```

```
catch(int){
cout<<"file isn't opening!"<<endl;
}
}

void ReadFromFile(){
Book obj;
ifstream read;
read.open("books.dat");
try{
if(!read){
throw 1;
}
read.read(reinterpret_cast<char*>(&obj),sizeof(obj));
while(!read.eof()){
cout<<"title: "<<obj.title<<endl;
cout<<"author: "<<obj.author<<endl;
cout<<"ISBN: "<<obj.ISBN<<endl;
cout<<"status: "<<obj.isavailable<<endl;
read.read(reinterpret_cast<char*>(&obj),sizeof(obj));
}
read.close();
}
catch(int){
cout<<"file isn't opening!"<<endl;
}
}
int main() {
cout<<"NAME: S.M.IRTIZA | ID: 22K-4638"<<endl;
Book book1,book2;
book1.Input();
ifstream f("books.dat", ios::binary);
f.seekg(sizeof(book1), ios_base::beg);
streampos position = f.tellg();
cout<<"current pointer position in file: "<<position<<endl;

AddToFile(book1);

f.seekg(sizeof(book1), ios_base::cur);
position = f.tellg();
cout<<"updated pointer position in file: "<<position<<endl;

ReadFromFile();
f.seekg(sizeof(book1), ios_base::end);
position = f.tellg();
cout<<"EOF pointer position: "<<position<<endl;
f.close();
return 0;
}
```

OUTPUT:


NAME: S.M.IRTIZA | ID: 22K-4638
title: abc
author: xyz
ISBN: 000
status[true=1/false=0]: 0
current pointer position in file: 226
data entered successfully!
updated pointer position in file: 452
title: charlie and chocolate factory
author: willy wonka
ISBN: 0021
status: 1
title: black hole
author: stephen hawkins
ISBN: 00121
status: 1
title: abc
author: xyz
ISBN: 000
status: 0
EOF pointer position: 904

QUESTION # 02:

CODE:

```
//NAME: S.M.IRTIZA | ID: 22K-4638
#include <iostream>
#include <fstream>
#include <string>

template <typename T>
void FileHandler(const std::string& filename, T& data, std::ios_base::openmode mode,
std::streampos position = 0) {
std::fstream file(filename, mode);
if (!file.is_open()) {
std::cerr << "Failed to open file " << filename << std::endl;
return;
}

if (position >= 0) {
file.seekg(position, std::ios::beg);
}

if (mode & std::ios::in) {
file.read(reinterpret_cast<char*>(&data), sizeof(data));
} else if (mode & std::ios::out || mode & std::ios::app) {
file.write(reinterpret_cast<const char*>(&data), sizeof(data));
}

file.close();
}

struct Book {
char title[50];
char author[50];
char isbn[50];
};

int main() {
std::cout<<"NAME: S.M.IRTIZA | ID: 22K-4638"<<std::endl;
std::string booksFile = "book.dat";
std::ios_base::openmode mode = std::ios::out | std::ios::binary;

try {
// Write two books to the file
Book book1{"The Lord of the Rings", "J.R.R. Tolkien", "9780618640157"};
Book book2{"To Kill a Mockingbird", "Harper Lee", "9780061120084"};

FileHandler(booksFile, book1, mode);
FileHandler(booksFile, book2, mode);
```

```
// Read the first book from the file and print its information
Book book3;
mode = std::ios::in | std::ios::binary;
FileHandler(booksFile, book3, mode, 0);

std::cout << "Title: " << book3.title << std::endl;
std::cout << "Author: " << book3.author << std::endl;
std::cout << "ISBN: " << book3.isbn << std::endl;
} catch (const std::ios_base::failure& e) {
std::cerr << "I/O error: " << e.what() << std::endl;
} catch (const std::bad_alloc& e) {
std::cerr << "Memory allocation error: " << e.what() << std::endl;
} catch (const std::bad_cast& e) {
std::cerr << "Invalid cast error: " << e.what() << std::endl;
} catch (const std::exception& e) {
std::cerr << "Exception occurred: " << e.what() << std::endl;
}

return 0;
}
```

OUTPUT:

```
NAME: S.M.IRTIZA | ID: 22K-4638
Title: To Kill a Mockingbird
Author: Harper Lee
ISBN: 9780061120084
```

QUESTION # 03:

You are tasked with building a program that can manage an inventory of products using binary files in C++. The program should have the following functionalities: 1. Add a new product to the inventory 2. Display all products in the inventory 3. Search for a product by name or ID 4. Edit an existing product in the inventory 5. Delete an existing product from the inventory 6. Calculate the total value of the inventory Each product should have the following attributes: • Name (string) • ID (integer) • Price (double) • Quantity (integer) The program should be menu-driven, where the user can select the desired functionality by entering a corresponding number. The program should handle invalid user input and use exception handling to avoid program crashes. The program should also use binary files to store and retrieve the inventory data. The program should support both formatted and unformatted input/output operations. Additionally, the program should allow the user to specify the file access mode, i.e., whether the file should be opened for reading, writing, or both. Finally, the program should use seek and tell functions to navigate through the binary file and edit/delete specific products. Your task is to implement the program using C++ and the following concepts: • Binary file input/output with modes and seek/tell functions • Template functions and classes

CODE:

```
//NAME: S.M.IRTIZA | ID:22K-4638
#include <iostream>
#include <fstream>
#include <cstring>
#include <exception>
#include <iomanip>

using namespace std;

// Product struct to store product attributes
struct Product {
char name[50];
int id;
double price;
int quantity;
};

// Inventory class to manage products
class Inventory {
private:
fstream file;
public:
Inventory() {}

// Opens the file with specified access mode
void open(const char* filename, ios_base::openmode mode) {
file.open(filename, mode | ios::binary);
if (!file) {
throw runtime_error("Unable to open file");
}
}

// Closes the file
```

```
void close() {
file.close();
}

// Adds a new product to the inventory
void addProduct(Product product) {
file.seekp(0, ios::end);
file.write(reinterpret_cast<const char*>(&product), sizeof(Product));
}

// Displays all products in the inventory
void displayAllProducts() {
Product product;
file.seekg(0, ios::beg);
while (file.read(reinterpret_cast<char*>(&product), sizeof(Product))) {
cout << "ID: " << product.id << endl;
cout << "Name: " << product.name << endl;
cout << "Price: " << fixed << setprecision(2) << product.price << endl;
cout << "Quantity: " << product.quantity << endl << endl;
}
}

// Searches for a product by name or ID
void searchProduct() {
int searchChoice;
cout << "Search by: " << endl;
cout << "1. ID" << endl;
cout << "2. Name" << endl;
cout << "Enter your choice: ";
cin >> searchChoice;

if (searchChoice == 1) {
int id;
cout << "Enter product ID: ";
cin >> id;

Product product;
bool found = false;
file.seekg(0, ios::beg);
while (file.read(reinterpret_cast<char*>(&product), sizeof(Product))) {
if (product.id == id) {
found = true;
cout << "ID: " << product.id << endl;
cout << "Name: " << product.name << endl;
cout << "Price: " << fixed << setprecision(2) << product.price << endl;
cout << "Quantity: " << product.quantity << endl << endl;
break;
}
}
```

```
if (!found) {
cout << "Product not found" << endl;
}
} else if (searchChoice == 2) {
char name[50];
cout << "Enter product name: ";
cin.ignore();
cin.getline(name, 50);

Product product;
bool found = false;
file.seekg(0, ios::beg);
while (file.read(reinterpret_cast<char*>(&product), sizeof(Product))) {
if (strcmp(product.name, name) == 0) {
found = true;
cout << "ID: " << product.id << endl;
cout << "Name: " << product.name << endl;
cout << "Price: " << fixed << setprecision(2) << product.price << endl;
cout << "Quantity: " << product.quantity << endl << endl;
}
}
if (!found) {
cout << "Product not found" << endl;
}
} else {
cout << "Invalid input" << endl;
}
}

// Edits an existing product in the

void editProduct(char* fileName, int id, Product newData)
{
std::fstream file(fileName, std::ios::binary | std::ios::in | std::ios::out);
if (!file.is_open())
{
std::cerr << "Error opening file!\n";
return;
}
file.seekg(0, std::ios::end);
int fileSize = file.tellg();
int numProducts = fileSize / sizeof(Product);
file.seekg(0, std::ios::beg);

bool found = false;
for (int i = 0; i < numProducts; i++)
{
Product p;
file.read(reinterpret_cast<char*>(&p), sizeof(Product));
```

```
if (p.id == id)
{
found = true;
file.seekp(i * sizeof(Product), std::ios::beg);
file.write(reinterpret_cast<const char*>(&newData), sizeof(Product));
std::cout << "Product with ID " << id << " updated successfully!\n";
break;
}
}

if (!found)
{
std::cerr << "Product with ID " << id << " not found!\n";
}

file.close();
}


template <class T>
void deleteProduct(char* fileName, int id)
{
std::fstream file(fileName, std::ios::binary | std::ios::in | std::ios::out);
if (!file.is_open())
{
std::cerr << "Error opening file!\n";
return;
}
file.seekg(0, std::ios::end);
int fileSize = file.tellg();
int numProducts = fileSize / sizeof(Product);
file.seekg(0, std::ios::beg);

bool found = false;
for (int i = 0; i < numProducts; i++)
{
Product p;
file.read(reinterpret_cast<char*>(&p), sizeof(Product));
if (p.id == id)
{
found = true;
file.seekp(i * sizeof(Product), std::ios::beg);
Product emptyProduct;
file.write(reinterpret_cast<const char*>(&emptyProduct), sizeof(Product));
std::cout << "Product with ID " << id << " deleted successfully!\n";
break;
}
}
```

```
if (!found)
{
std::cerr << "Product with ID " << id << " not found!\n";
}

file.close();
}

// Function to calculate the total value of the inventory
double calTotalValue(fstream& file) {
// Reset the file pointer to the beginning
file.clear();
file.seekg(0, ios::beg);

double totalValue = 0;
Product product;

// Read the product data from the file and calculate the total value
while (file.read(reinterpret_cast<char*>(&product), sizeof(product))) {
totalValue += product.price * product.quantity;
}

return totalValue;
}

};
```

QUESTION # 04

Write a C++ program that implements a template class Person that represents a person with a name and an age. The class should have a constructor that takes a name and an age and sets the member variables accordingly. The class should also have member functions getName and getAge that return the name and age of the person, respectively. Create two additional classes that inherit from Person: • Student: represents a student with a major and a GPA. The class should have a constructor that takes a name, an age, a major, and a GPA, and sets the member variables accordingly. The class should also have member functions getMajor and getGPA that return the major and GPA of the student, respectively. • Employee: represents an employee with a job title and a salary. The class should have a constructor that takes a name, an age, a job title, and a salary, and sets the member variables accordingly. The class should also have member functions getJobTitle and getSalary that return the job title and salary of the employee, respectively. Create a template class Pair that represents a pair of two elements of different data types. The class should have a constructor that takes two arguments of different data types and sets the member variables accordingly. The class should also have member functions getFirst and getSecond that return the first and second elements of the pair, respectively. Finally, create a function printInfo that takes a Person, a Student, or an Employee, and prints their name and age, as well as any additional information that is specific to the derived class (i.e., major and GPA for a Student, or job title and salary for an Employee). Your program should demonstrate the use of the Pair template class and the printInfo function by creating instances of Person, Student, and Employee, creating instances of Pair that store different combinations of data types, and calling printInfo with each instance.

CODE:

```
//NAME: S.M.IRTIZA | ID: 22K-4638
#include <iostream>
#include <string>

using namespace std;

template<typename T1, typename T2>
class Pair {
private:
T1 first;
T2 second;
public:
Pair(T1 f, T2 s) : first(f), second(s) {}
T1 getFirst() { return first; }
T2 getSecond() { return second; }
};

class Person {
private:
string name;
int age;
public:
Person(string n, int a) : name(n), age(a) {}
string getName() { return name; }
int getAge() { return age; }
};
```

```
class Student : public Person {
private:
string major;
double gpa;
public:
Student(string n, int a, string m, double g) : Person(n, a), major(m), gpa(g) {}
string getMajor() { return major; }
double getGPA() { return gpa; }
};

class Employee : public Person {
private:
string jobTitle;
double salary;
public:
Employee(string n, int a, string j, double s) : Person(n, a), jobTitle(j), salary(s) {}
string getJobTitle() { return jobTitle; }
double getSalary() { return salary; }
};

void printInfo(Person& p) {
cout << "Name: " << p.getName() << endl;
cout << "Age: " << p.getAge() << endl;
if (typeid(p) == typeid(Student)) {
Student& s = static_cast<Student&>(p);
cout << "Major: " << s.getMajor() << endl;
cout << "GPA: " << s.getGPA() << endl;
}
if (typeid(p) == typeid(Employee)) {
Employee& e = static_cast<Employee&>(p);
cout << "Job Title: " << e.getJobTitle() << endl;
cout << "Salary: " << e.getSalary() << endl;
}
cout << endl;
}


int main() {
cout<<"NAME: S.M.IRTIZA | ID: 22K-4638"<<endl;
Person p("John Smith", 30);
Student s("Jane Doe", 20, "Computer Science", 3.5);
Employee e("Bob Johnson", 45, "Manager", 50000);

Pair<int, string> p1(1, "hello");
Pair<string, double> p2("world", 3.14);

printInfo(p);
printInfo(s);
printInfo(e);
```

```
cout << "Pair 1: " << p1.getFirst() << ", " << p1.getSecond() << endl;
cout << "Pair 2: " << p2.getFirst() << ", " << p2.getSecond() << endl;

return 0;
}
```

OUTPUT: