Worksheet 2

Task 1

1. Which of the following statements is NOT true about inheritance in C++?

a) Derived class inherits public and protected members of the base class.

b) Private members of the base class are not accessible by the derived class directly.

c) Derived class can access public members of the base class using the scope

resolution operator.

d) A class can inherit from multiple base classes

Ans= C

2. \_\_\_\_\_\_\_\_\_\_\_is called automatically each time an object is created.

a. compiler

b. builder

c. constructor

d. destructor

Ans= C

3. Which of the following statements is true ?

a. A friendfunction can declare itself to be a friend of a class.

b. A friendfunction can declare itself to be a friend of another function.

c. Both of the above are true.

d. None of the above is true.

Ans= A

4. A constructor is defined as Machine(int = 352, int = 2, double = 1400.00);. Which of the

following is a legal statement that uses the constructor?

a. Machine myMachine(128);

b. Machine myMachine(1234.56);

c. Machine myMachine(256, 899.99);

d. All of the above are legal.

Ans= D

5. A function that has been declared to be a friend of a class has access to the \_\_\_\_\_\_\_\_\_\_\_data

in the class.

a. public

b. private

c. both of the above

d. none of the above

Ans= B

6. When an object goes out of scope, a(n) \_\_\_\_\_\_\_\_\_\_\_is called

automatically.

a. destructor

b. constructor

c. overloaded function

d. operating system error message

Ans= A

7. Which of the following is a legal example of the way overloading is used in C++?

a. creating any two functions for a class

b. creating two constructors for a class

c. creating two destructors for a class

d. all of the above

Ans= B

8. The primary advantage to overloading functions is \_\_\_\_\_\_\_\_\_\_\_.

a. you can use one function name for every operation you want to perform with a class

b. you can use one class name for many types of items

c. you can use one function name for similar operations, regardless of the data types

involved

d. you do not have to write separate code for the bodies of functions that perform

similar tasks

Ans= C

9. Whenever a class contains two constructors, the constructors are

\_\_\_\_\_\_\_\_\_\_\_.

a. default constructors

b. destructors

c. overloaded

d. static

Ans= C

10. A constructor has been defined as Box(int = 7);. Which of the following constructors

could coexist with the defined constructor without any possible ambiguity?

a. Box (int);

b. Box ();

c. Box (int = 10, int = 18);

d. Box (int, double);

Ans=(A,B,D)

Task 2

#include <iostream>

using namespace std;

class Rect

{

private:

int width;

int height;

public:

// Default Constructor

Rect()

{

width = 0;

height = 0;

}

// Parameterized Constructor

Rect(int w, int h)

{

width = w;

height = h;

}

// Input values from terminal

void getInput()

{

cout << "Enter height of rectangle: ";

cin >> height;

cout << "Enter width of rectangle: ";

cin >> width;

}

void getDisplay()

{

cout << "Height of rectangle: " << height << endl;

cout << "Width of rectangle: " << width << endl;

cout << "The Area of rectangle is: " << height \* width << endl;

}

// Overloading + operator

int operator+(Rect r1)

{

int a1 = height \* width;

int a2 = r1.height \* r1.width;

return a1 + a2;

}

// Overloading \* operator

int operator\*(int c)

{

return height \* width \* c;

}

// Check if the rectangle is a square or not

int CheckSq()

{

if (height == width)

return 1;

else

return 0;

}

};

int main()

{

Rect r1(4, 4);

Rect r2;

r2.getInput();

cout << "The given Rectangle 1 values:" << endl;

r1.getDisplay();

if (r1.CheckSq())

cout << "Rectangle 1 is a square." << endl

<< endl;

else

cout << "Rectangle 1 is not a square." << endl

<< endl;

cout << "The given Rectangle 2 values:" << endl;

r2.getDisplay();

if (r2.CheckSq())

cout << "Rectangle 2 is a square." << endl

<< endl;

else

cout << "Rectangle 2 is not a square." << endl

<< endl;

int sumArea = r1 + r2;

cout << "Sum of areas of both rectangle is: " << sumArea << endl;

int temp = r2 \* 4;

cout << "Multiplying area of rectangle by 4: " << temp << endl;

}

Task 3

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

const int BOOKS = 10;

const int LENGTH = 150;

struct Book {

char title[LENGTH];

};

void writeBooksToFile(const char\* filename, const Book books[], int numBooks) {

ofstream outFile(filename, ios::binary);

if (!outFile) { // Corrected the condition to properly check if the file did not open

cerr << "Error opening the file for writing." << endl;

return;

}

outFile.write(reinterpret\_cast<const char\*>(books), sizeof(Book) \* numBooks);

outFile.close();

}

bool isTitleInFile(const char\* filename, const char\* searchBook) {

ifstream inFile(filename, ios::binary);

if (!inFile) { // Corrected the condition to properly check if the file did not open

cerr << "Error opening the file for reading." << endl;

return false;

}

Book book;

while (inFile.read(reinterpret\_cast<char\*>(&book), sizeof(Book))) {

if (strcmp(book.title, searchBook) == 0) {

inFile.close();

return true;

}

}

inFile.close();

return false;

}

int main() {

Book books[BOOKS];

cout << "Type in the title of the 10 books:" << endl;

for (int s = 0; s < BOOKS; ++s) {

cout << "Book " << s + 1 << ": ";

cin.getline(books[s].title, LENGTH);

}

writeBooksToFile("books.dat", books, BOOKS);

cout << "To search, type the name of a book: ";

char searchBook[LENGTH];

cin.getline(searchBook, LENGTH);

if (isTitleInFile("books.dat", searchBook)) {

cout << "The book \"" << searchBook << "\" is on file." << endl;

} else {

cout << "The book \"" << searchBook << "\" is not on file." << endl;

}

return 0;

}