VERSION - A

**OOP244: Introduction to Object Oriented Programming**

**Professor**: Bradly Hoover

Sections A, B 24 June 2016

Duration of test: 100 minutes

This test includes 7 pages and 6 sections of questions for a total of 40 marks. You are responsible for ensuring that your copy of the paper is complete. Bring any discrepancy to the attention of your invigilator. **Include your information on the top of ALL pages**.

**Special instructions**: You are allowed to bring **one double-sided HANDWRITTEN reference sheet on 8-1/2 x 11 inch paper,** an English language dictionary, a calculator and a rubber duck. No electronic references, electronic devices, textbooks or other assistance is allowed. Answer sections 1 and 2 on these papers and sections 3 – 6 in the exam booklet. Hand in this paper along with your reference sheet and any additional exam booklets at the end of the allocated time. Label each page and booklet with your name and student number (including the reference sheet). If you write the test in pencil, the test will not be remarked after it has been returned and the grade given will be final. The value of each section is given at the beginning of the corresponding section.

**Section #1 – General C++ and Object Orientated Programming [4 Marks]**

1. [1 mark] Explain the purpose of the following preprocessor directives that are found in header files.

#ifndef HEADER\_FILE\_NAME\_H

#define HEADER\_FILE\_NAME\_H

// code

#endif

The purpose of the preprocessor directive is to prevent the header file from being included twice, thus breaking the multiple definition rule.

1. [1 mark] What are the 3 stages of the compilation process?

Preprocessing, Compiling, Linking

1. [1 mark] What are the 3 core concepts in Object Orientated Programming?

Encapsulation, Inheritance, Polymorphism

1. [1 marks] Explain why the output of the program is the following.

Program Output

|  |  |
| --- | --- |
| #include <iostream>  using namespace std;  int main() {  int x = 10 ;  for(int i = 0; i < 3; i++ ){  int x = i + 1;  cout << x << endl;  }  cout << x << endl;  } | 1  2  3  10 |

The reason the output of the program is what it is, is because the x defined in the for loop shadows the x defined outside of the for loop. This means that the program will use the most local copy when there are multiple variables that share the same name.

**Section #2 – C++ Input/Output [6 marks]**

1. [3 marks] Fill in the code to get the required output.

Output

|  |
| --- |
| abcdefg  Z------ |
| #include <iostream>  #include < iostream >  using namespace std;  int main(){  cout << “abcdefg“ <<\_\_\_\_endl\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  cout <<\_\_\_left\_\_\_\_\_\_<<\_\_\_\_setfill(‘-‘)\_\_\_\_<<\_\_\_\_\_setw(7)\_\_\_<< “Z”;  } |

1. [3 marks] Add the code required to retrieve input data from the user and store them in the local variables declared. Assume that the user will input this information on one line with a space between the requested values. Also assume that the user will enter valid input.

|  |
| --- |
| #include <iostream>  int main(){  char name[30];  int num;  std::cout << “Enter a student name and a student number\n”;  // add your code here  std::cin >> name >> num;  std::cout << “Student Name: ” << name << “ Student Number: “ << num;  } |

**Section #3 – Class Design [8 marks]**

book.h

|  |
| --- |
| #include <cstring>  using namespace std;  **class** Book {  **char** \_author[61];  **char** \_title[31];  **int** \_num\_pages;  public:  Book(**const char\*** author, **const char\*** title, **int** pages);  // getters/query functions for private data members  **const char\*** getAuthor() const { return \_author; };  **const char\*** getTitle() const { return \_title; };  **int** getNumOfPages() const { return \_num\_pages; };  // setters/modifier functions for private data members  **const char\*** setAuthor(**const char\*** a){ strcpy(\_author, a); };  **const char\*** setTitle(**const char\*** t) { strcpy(\_title, t); };  **int** setNumOfPages(**int** n) { \_num\_pages = n; };  }; |

Use the Book class declared above for the following questions.

1. [4 marks] Implement the constructor declared above. If all of the string arguments are not null pointers and are not empty, and the number of pages passed in positive, the constructor should initialize all of the class variables. Otherwise set the object to a safe empty state. Assume the string arguments are of a valid length.

Book::Book(const char\* author, const char\* title, int pages){

\_author[0] = \_title[0] = ‘\0’;

\_pages = -1;

if(author && title && author[0] != ‘\0’ && title[0] != ‘\0’ && pages > 0){

strcpy(\_author, author);

strcpy(\_title, title);

\_pages = pages;

}

}

1. [4 marks] Implement a helper operator to overload the “==” operator. This operator should return true if the Author and title of the two books being compared are the same.

bool operator=(const Book& lhs, const Book& rhs){

return (strcmp(\_lhs.getAuthor(), \_rhs.getAuthor()) == 0) &&

(strcmp(\_lhs.getTitle(), \_rhs.getTitle()) == 0);

}

**Section #4 – Classes with Resources [14 marks]**

library.h

|  |
| --- |
| #include “book.h”  class Library{  char \_name[61];  int \_year;  Book\* \_list\_of\_books;  int \_num\_books;  public:  int getTotalPages() const;  void addBook(const Book&);  Library(const char\* n, int y);  Library(const Library& other);  Library& operator=(const Library& other);  ~Library();  }; |

test.cpp

|  |
| --- |
| #include <iostream>  #include “library.h”  int main(){  Library library(“Toronto Public Library”, 1884);  Book b1;  b1.setTitle(“The Stand”);  b1.setAuthor(“Stephen King”);  b1.setNumOfPages(986);  Book b2;  B2.setTitle(“The Way of Kings”);  B2.setAuthor(“Brandon Sanderson”);  B2.setNumOfPages(1004);  library.addBook(b1);  library.addBook(b2);  std::cout << “The total number of pages in the library is “  << library.getTotalPages() << “.” << std::endl;  } |

Use the Library class from above to complete the following questions. Assume that the Book class in section 3 has been fully implemented.

1. [3 marks] Implement the getTotalPages() function in the library class. It must return the total number of pages from all of the books in the library combined.

int Library::getTotalPages(){

int total = 0;

for(int i = 0; i < \_num\_books; ++i){

total += \_list\_of\_books[i].getNumOfPages();

}

return total;

}

1. [3 marks] Implement the destructor for the library class.

Library::~Library(){

delete [] \_list\_of\_books;

}

1. [5 marks] Implement the assignment operator for the Library class.

Library& Library::operator=(const Library& other){

if(this != &other){

\_year = other.\_year;

Strcpy(\_name, other.\_name);

\_num\_books = other.\_num\_books;

delete [] \_list\_of\_books;

\_list\_of\_books = nullptr;

if(other.\_list\_of\_books){

\_list\_of\_books = new Book[\_num\_books];

for(int i = 0; i < \_num\_books; ++i){

\_list\_of\_books[i] = other\_list\_of\_books[i];

}

}

}

return \*this;

}

1. [5 marks] Implement the copy operator for the Library class.

Library::Library(const Library& other){

\_year = other.\_year;

Strcpy(\_name, other.\_name);

\_num\_books = other.\_num\_books;

\_list\_of\_books = nullptr;

if(other.\_list\_of\_books){

\_list\_of\_books = new Book[\_num\_books];

for(int i = 0; i < \_num\_books; ++i){

\_list\_of\_books[i] = other\_list\_of\_books[i];

}

}

}

1. [3 marks] The compiler generates a default no-argument destructor with an empty body. Explain why a programmer would want to provide their own version of the destructor.

You should implement your own version of the no argument destructor because if your class manages resources, you should ensure that the resources are properly handles upon destruction. Recourses that are improperly handles upon destruction will cause a memory leak or improperly released resources.

1. [3 marks] The compiler generates a default no-argument constructor with an empty body. Explain why a programmer would want to provide their own version of the destructor.

You should implement your own version of the no argument constructor to ensure that when an object is constructed using the no argument constructor that the object will be put in a safe state. If you do not implement the constructor, the data embers in the class will hold garbage values, possibly causing the program cause an error when the object is first accessed.

**Section #6 – Code Walkthrough [8 marks]**

|  |
| --- |
| class Foo {  int number;  char string[32];  public:  void set(int a, int b, const char\* str){  number = a \* b;  strcpy(string, str);  }  void update(){  if(number >= 20) {  strcat(string, " juice.");  number \*= 2;  } else if(number >= 10) {  strcat(string, " candy.");  number \*= 3;  } else {  strcat(string, " jam.");  number \*= 4;  }  }  void output() const{  cout << "Number is: " << number << " and string is: " << string  << endl;  }  };  void updateAll(Foo \*a, Foo &b, Foo c) {  a->update();  b.update();  c.update();  }  int main() {  Foo x, y, z;  x.set(3, 5, "Grape");  y.set(4, 6, "Orange");  z.set(2, 4, "Strawberry");  updateAll(&x, y, z);  x.output();  y.output();  z.output();  return 0;  } |

State the output of the above program. Show rough work.

Number is: 45 and string is: Grape Candy

Number is: 48 and string is: Orange Juice

Number is: 8 and string is: Strawberry