CISC 181 Final Exam

Print Name:

Signature:

1. (40 pts) Before each concept in the left column write the number of its definition from the right column. For ease of reference the items in the left-hand column are listed alphabetically.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract class</td>
<td>1. The ability for objects of different classes related by inheritance to respond differently to the same message. I.e., that is to member functions with the same name.</td>
</tr>
<tr>
<td>actual argument</td>
<td>2. A data object that is used with insertion and extraction operators as if it were a value to be inserted into a stream or a data object whose value is to be extracted from the stream. In fact, however, it causes a specified operation to be performed on the stream which may or may not cause insertions or extractions.</td>
</tr>
<tr>
<td>address</td>
<td>3. A class for which it is not possible to instantiate an object of the class. Often serves as a base class for an inheritance hierarchy.</td>
</tr>
<tr>
<td>constructor</td>
<td>4. A file containing (primarily) declarations that are to be included in one or more source files.</td>
</tr>
<tr>
<td>copy constructor</td>
<td>5. A C++ function type that is used to implement polymorphism.</td>
</tr>
<tr>
<td>data abstraction</td>
<td>6. A programming methodology that models real world objects with software classes.</td>
</tr>
<tr>
<td>default argument</td>
<td>7. The operator :: that is used to indicate the class to which a member function belongs when it is defined outside the class definition.</td>
</tr>
<tr>
<td>derived class</td>
<td>8. A relationship between classes in which an object of a derived class can be considered as an object of the base class.</td>
</tr>
<tr>
<td>destructor</td>
<td>9. A function or operator declared in a class and not declared as a friend.</td>
</tr>
<tr>
<td>dynamic memory allocation</td>
<td>10. A C++ language feature that enables a programmer to specify, with a single code segment, an entire range of related, overloaded, functions or an entire range of related classes.</td>
</tr>
<tr>
<td>formal argument</td>
<td>11. A variable that is implicitly declared in every member function of a class.</td>
</tr>
<tr>
<td>header file</td>
<td>12. The operator * when used to access a value referred to by a pointer.</td>
</tr>
<tr>
<td>indirection or</td>
<td>13. A mechanism that allows C++ operators to be used with programmer defined types.</td>
</tr>
<tr>
<td>dereferencing operator</td>
<td></td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>“is a” relationship</td>
<td>14. A function invoked when creating an instance of a class.</td>
</tr>
<tr>
<td>member function</td>
<td>15. A property of class objects whereby the internal structure of an object is hidden from the rest of the program, which can interact with the object only through its public part.</td>
</tr>
<tr>
<td>object-oriented programming</td>
<td>16. An argument value that is specified in a function declaration or prototype and is used if the corresponding actual argument is omitted when the function is called.</td>
</tr>
<tr>
<td>operator overloading</td>
<td>17. A variable declared in a function heading that is initialized with the value of the corresponding actual argument.</td>
</tr>
<tr>
<td>pointer</td>
<td>18. The means by which data objects can be explicitly created as they are needed and remain in existence until they are explicitly deallocated.</td>
</tr>
<tr>
<td>polymorphism</td>
<td>19. The method that initializes a new class object from an existing object of the same class.</td>
</tr>
<tr>
<td>private member</td>
<td>20. The address of a data object of a particular type.</td>
</tr>
<tr>
<td>scope</td>
<td>21. The portion of a program text in which an identifier is accessible.</td>
</tr>
<tr>
<td>scope resolution operator</td>
<td>22. An expression whose value is passed to a function when the function is called.</td>
</tr>
<tr>
<td>stream manipulator</td>
<td>23. A function that is called implicitly to deallocate dynamically allocated storage when a class object goes out of scope.</td>
</tr>
<tr>
<td>templates</td>
<td>24. A value that indicates where an item is stored in memory, much like a post office box number.</td>
</tr>
<tr>
<td>this</td>
<td>25. A class member that is accessible only to a class’ member and friend functions.</td>
</tr>
<tr>
<td>virtual function</td>
<td>26. A class that is inherited from another class.</td>
</tr>
</tbody>
</table>
2. (10 pts) What is printed by the following program? Draw a memory map that illustrates the contents of the various variables used in the program. Indicate the values of pointer variables with arrows pointing to the variables whose address they contain. As the values of the variables change, mark through the old values and enter the new ones thereby maintaining a history of all the values assigned to the variables.

```c
#include <stream.h>

void swap(int* &p1, int* &p2);

main()
{
    int y = 11, z;
    int* r, *s = &z;
    int& a = y;

    r = &y;
    z = 33;

    cout << a << " " << y << " " << z << endl;
    cout << *r << " " << *s << endl;

    *s = 44;
    a = 55;

    cout << a << " " << y << " " << z << endl;
    swap(r,s);
    cout << *r << " " << *s << endl;
}

void swap(int* &p1, int* &p2)
{
    int* temp = p1;
    p1 = p2;
    p2 = temp;
}
```

3. (5 pts) Using labeled boxes for classes and arrows to indicate inheritance, arrange the following classes in a natural inheritance hierarchy: Vehicle, Car, Truck, PickupTruck, DumpTruck.

4. (5 pts) What are the natural relationships between

(a) a Person class and a Name class?

(b) a Person class and an Address class?

(c) a Student class and a Person class?
5. (10 pts) Study the following partial program and then fill-in the blanks at the bottom of the page.

class Graphics{
    public:
        Graphics( int rows = 50, int cols = 50 ); // Constructor
        ~Graphics(); // Destructor
        virtual void clearRect( int x, int y, int width, int height );
        virtual void drawBorder(); // Draw a border around grid
        virtual void drawLine( int x1, int y1, int x2, int y2 );
        virtual void drawString( char* str, int x, int y );
        virtual void show() const; // Display current grid
    protected:
        char** grid; // pointer to a 2-dim'l grid of chars
        int rows, // Number of rows in the grid
            cols; // Number of columns in the grid
};

Graphics::Graphics( int r, int c ) // Constructor
{
    rows = r; cols = c;
    this->grid = new char*[rows]; // Allocate pointers to each row
    for (int i = 0; i < rows; i++) // Allocate space for each row
    {
        grid[i] = new char[cols+1]; // Extra space for null char
        grid[i][cols] = '\0'; // Set last char of each row
    } // to the null char.

    int width = cols, height = rows;
    clearRect(0,0, width, height); // Clear the grid
}

main(){
    Graphics g; // Create a "canvas" g on which to paint

    g.drawBorder(); // Draw border around the canvas
}

(a) The type of the variable rows in the function Graphics() is ____________________.
(b) The type of the variable grid[i] in the function Graphics() is ____________________.
(c) The type of the variable grid[i][cols+1] in the function Graphics() is ________________.
(d) The type of the variable this in the function Graphics() is ____________________.
(e) The type of the variable g in the function main() is ____________________.
6. (a) (5 pts) Write a C++ function
   ```cpp
   int strlen(const char string[])  
   ```
   that computes the length of a string. Be sure that it works for the empty string, i.e., a string of length zero.
   (b) (10 pts) Use the `strlen()` function to write a function
   ```cpp
   bool palindrome(const char string[]) 
   ```
   that returns `true` if `string` is a palindrome; `false` otherwise. Recall that a palindrome is a string that reads the same backwards and forwards, e.g. “noon”, “bob”, or “able was I ere I saw elba”. The empty string is also a palindrome.

7. (15 pts) Consider the following class declaration.

   ```cpp
   class Name{
   
   public:
   
   Name(const char* first = "", const char* last = "");
   ~Name();
   Name& operator=( const Name& rhs);
   
   private:
   
   char* firstName;
   char* lastName;
   
   };
   ```

   Give the implementation of the three methods whose prototypes are given in the class declaration.