New York University – Tandon School of Engineering
Brooklyn, NY 11201

CS2124
FINAL Exam – 30 June 2023

Name: _____

Email:_____

This exam is **CLOSED BOOK**
You have 2 hours to complete this exam

PLEASE DO NOT WRITE ON THE BACK OF ANY PAGE!

Note that I have omitted any #includes or "`using namespace std;`" statements in all questions, in order to save space and to save your time thinking about them. You may assume that all such statements that are needed are present. And you don't have to write them either!!!

Please, read all questions carefully! Answering the short-answer questions, in particular, requires that you read and understand the programs shown. But you do need to carefully read them if you are going to understand them.

Anyone found cheating on this exam will immediately fail. Anyone who is found writing after time has been called will also fail. Do not open this test booklet until you are instructed to do so. If you have a question, please ask only the proctor of the exam!

1. (5 pts) What would be output by the program below?

```
void incr(int* num, int n_times) {
  for (int i = 0; i < n_times; i++) num = num + 2;
}

int main() {
  int val = 2;
  incr(&val, 3);
  cout << val;
}
```

O 8
O 4
O 2
O The program compiles and crashes when it runs
O 5
O 6
O The program fails to compile
O None of these

2. (5 pts) Provide a single line of code which creates a **`Cookie`** object on the heap and stores its location in a pointer named **`snack`** which has already been created. The **`Cookie`** constructor takes two parameters, the first is an int, the size of the cookie, and the second is a double, the temperature of the cookie.

3. (3 pts) A class which contains one or more pure virtual functions is best referred to as

O An abstract class
O A derived class
O An inherited class
O An interface class
O A sliced class
O A base class
O A polymorphic class
O An overloaded class

4. (3 pts) The correct return type of the post increment operator for the **`Class`** class would be what?

O **`Class`**
O **`Class&`**
O **`void`**
O **`ostream`**

5. (3 pts) A class which does not overload the assignment operator:

    - O Cannot be copied (x=y doesn't work)
    - O Accepts a shallow copy
    - O Performs a deep copy automatically
    - O Can be copied only if it is used as a base class in inheritance.

6. (3 pts) Assuming **`Base`** and **`Derived`** classes are designed appropriately, and **`basePtr`** and **`derivedPtr`** are pointing to objects of their respective classes, which of the following lines of code will result in polymorphism?

    - O **`derivedPtr = basePtr`**
    - O **`basePtr = derivedPtr`**
    - O **`*derivedPtr = *basePtr`**
    - O **`*basePtr = *derivedPtr`**

7. (3 pts) If a class is templated and a method of that class is defined outside of the declaration of the class,

    - O the method must be templated
    - O the class cannot be instantiated
    - O the method must be overloaded
    - O the method must accept the templated datatype as a parameter.
    - O All of the above
    - O None of the above

8. (5 pts) What is the output of running the following program (please **read it carefully**, there will be no partial credit!):

```
#include <iostream>
using namespace std;
class Embedded {
public:
      Embedded() {cout << 1;}
      ~Embedded() {cout << 2;}
};
class Base {
public:
      Base() {cout << 3;}
      ~Base() {cout << 4;}
private:
      Embedded embeddedInBase;
};

class Derived : public Base {
public:
      Derived() {cout << 5;}
      ~Derived() {cout << 6;}
};
int main() {
      Derived a;
}
```

O **1 2 3 4 5 6**
O **1 3 5 2 4 6**
O **1 5 3 2 4 6**
O **1 5 3 4 2 6**
O **1 5 3 6 4 2**
O **1 3 5 6 4 2**
O **1 3 5 4 2 6**
O **None of the above**
O **More than one of the above**

9. (15 pts) The **IceCreamCone** class below, stores integer IDs where each integer is used to represent a unique number indicating the flavor of each scoop of ice cream. Our company is changing and we are not going to store integers anymore, but we're still unsure of what data type we will store in the **IceCreamCone** class. Please rewrite the *entire* program shown below changing the class **IceCreamCone** into a templated class so that it can be used to hold types other than integers. Do not implement any additional methods. The function **main** should still create an **IceCreamCone** that holds integers for backward compatibility.

```
class IceCreamCone
      {
      private:
          int * scoopIDs;
          int numOfScoops;
      public:
          IceCreamCone() {numOfScoops=5; scoopIDs=new int[5];}
          void changeConeSize(int newsize);
          int& getScoopID(int index);
          int getConeSize() const;
          int min(int a, int b) {return (a<b)?a:b;}
      };

      void IceCreamCone::changeConeSize(int newsize)
      {
          int * temp = new int[newsize];
          for (int i=0; i<min(newsize, numOfScoops); i++)
              temp[i]=scoopIDs[i];
          delete[] scoopIDs;
          scoopIDs=temp;
          numOfScoops=newsize;
      }

      int& IceCreamCone::getScoopID(int index)
      {
          return scoopIDs[index];
      }

      int IceCreamCone::getConeSize() const
      {
          return numOfScoops;
      }

      int main()
      {
          IceCreamCone f;
          f.changeConeSize(10);
      }
```

10. (30 pts) Our customer is a Coffee Shop which has asked us to design a new system for placing orders. We will be implementing the classes for this system. At the coffee shop orders can be placed for **`IcedCoffee`** or **`HotCoffee`** only (we're keeping it simple). All coffee items have a size (an integer number of ounces), milk (a boolean), and sugar (a Boolean). **`IcedCoffees`** have an amount of ice (an integer, 5 as the default from 1-10) and **`HotCoffees`** have a temperature (a double, 195.5 as the default). Each order can contain one or more coffees including both hot and iced, possibly in the same order. In the future, we may add more types of Coffee to the order class (cappuccino anyone?) so your order class must accommodate all types of coffee, not just hot and iced.

    Ingredients are expensive, so you must provide a "cost" method in each class. The cost of an order is of course, the sum of all of the costs of all of the coffees in that order. Use the following table to determine the price of the order:

Iced coffee base cost: $3.00 (additional costs below)			**Hot coffee base cost: $2.50 (additional costs below)**	
Size>32	$1.00		Size>16	$1.00
Add Milk	$0.50		Add Milk	$0.25
Add Sugar	$0.25		Add Sugar	$0.15
Ice>5	$0.50		Temperature>210	$0.25

    You should design the above classes including the **`HotCoffee`**, **`IcedCoffee`** and **`Order`** class as well as any other classes you might need. Below is a sample **`main`** function to demonstrate how this works. Your code MUST work with this below example.