CMPS 222 Sample Final

1. (5 pts) Polymorphic functions differ from normal functions due to late binding of the function call to the function body. Define late binding and describe how it differs from normal function call linking done at compilation time (static binding).

2. (5 pts) How does exception handling differ from using an if-else statement to handle an error?

3. (5 pts) For the following operators, state whether the operator must be a friend function, should be a friend function or should be a member function:

   (a) The output operator, <<

   (b) The addition operator, +

   (c) The increment operator, ++

4. (5 pts) When you have a template class, why do all of the member functions have to be template functions, even if that member function does not access one of the template member variables?

5. (5 pts) When a child object is passed to a function that expects a parent object, a phenomena called slicing occurs. Define slicing and explain why slicing is necessary to convert a child object into a parent object.

6. (5 pts) For certain operators, the same operator symbol can have multiple meanings. For example, the minus sign (-) can mean either subtraction (a - b) or negation (-a). How do you distinguish between these two meanings when writing the operators for your class since they would both have the function name `operator-`?

7. (5 pts) Why does separate compilation not work with template classes? When answering, consider what happens when you declare a variable of the template class type, such as GenericList<int> iList.

8. (5 pts) When your class has a dynamic array as a member variable, certain features need to be coded in order for your class to work correctly. For each of the following features, state what would go **wrong** if you neglected to include that feature in your class:

   (a) Copy constructor

   (b) Destructor

9. (5 pts) For the following code, give the current state of memory at each indicated point.

```
int main() {
  int *p1 = NULL, *p2 = NULL;
  int v1 = 15;
  int arr[] = { 5, 6, 3 };

// Fill out the memory map for the current state of the program at this point
// Address:   80000    80004    80008    80012    80016    80020    80024
//           ----------------------------------------------------------------
//          |        |        |        |        |        |        |        |
// Value:   |        |        |        |        |        |        |        |
//          |________|________|________|________|________|________|________|
//
// Variable:    p1       p2       v1     arr[0]   arr[1]   arr[2]

  p2 = &v1;
  p1 = arr;

// Fill out the memory map for the current state of the program at this point
// Address:   80000    80004    80008    80012    80016    80020    80024
//           ----------------------------------------------------------------
//          |        |        |        |        |        |        |        |
// Value:   |        |        |        |        |        |        |        |
//          |________|________|________|________|________|________|________|
//
// Variable:    p1       p2       v1     arr[0]   arr[1]   arr[2]

  arr[3] = 9;
  *p2 = 20;
  p1++;
  *p1 = 10;

// Fill out the memory map for the current state of the program at this point
// Address:   80000    80004    80008    80012    80016    80020    80024
//           ----------------------------------------------------------------
//          |        |        |        |        |        |        |        |
// Value:   |        |        |        |        |        |        |        |
//          |________|________|________|________|________|________|________|
//
// Variable:    p1       p2       v1     arr[0]   arr[1]   arr[2]

  return 0;
}
```

10. (5 pts) You have the parent class Employee and the child class HourlyEmployee. Employee has the name and SSN of the employee while HourlyEmployee adds the hourly rate and hours worked of the employee. Answer the following questions for the following code snippet:

```
// The class definitions and remainder of class functions would be here
// but have been removed for sake of space.
void Employee::print_check() {
  printf("Name: %s\n", getName());
}
void HourlyEmployee::print_check() {
  Employee::print_check();
  printf("SSN: %s\n", getSSN());
  printf("Salary: $%.2f\n", getSalary());
}
int main() {
  HourlyEmployee sally;
  Employee *e;

  sally.setName("Sally Smith");
  sally.setSSN("999999999");
  sally.setHours(40);
  sally.setRate(10.00);

  e = &sally;
  e->print_check();

  return 0;
}
```

   (a) What would be printed out for $e \rightarrow print\_check()$ if $print\_check()$ were a normal function (static binding) that had been redefined in the child?

   (b) What would be printed out for $e \rightarrow print\_check()$ if $print\_check()$ were a polymorphic function, e.g. declared with the virtual keyword?

11. (10 pts) You have the parent class Employee as described in Question 10. You wish to define an additional child class called SalariedEmployee that will inherit from Employee publicly. The SalariedEmployee class will have the private member variable *salary*, a default constructor and the public member functions *getSalary*, *setSalary* and *print_check*. Write the definition for the SalariedEmployee class and the body of each of the member functions (either inline or normally). Invoke the parent class as needed, such as in the constructor or *print_check*. Assume *print_check* is polymorphic for this question.

12. (10 pts) You have the GenericList template class, as described in Homework 6, which stores a partially filled dynamic array. Add the following features to the given code (each feature is worth 2 points):

    - The `InvalidIndex` exception class.
    - A destructor that deallocates the array when it is not NULL.
    - A template version of the output operator that outputs each element separated by a space.
    - A specific-instance version of the output operator for GenericList<double> that outputs each element with 4 decimal places of precision, separated by a space.
    - An index operator that throws `InvalidIndex` when the requested index is not in use. Otherwise it returns the array element at that index.

    NOTE: This question continues on to the next page. Make sure to implement all the above functions in the space provided.

```
//=====================================================================
// Add exception class here
//=====================================================================


template <class T>
class GenericList {
  private:
    T *array;
    int cap;
    int count;
  public:
    GenericList() { array = NULL; cap = 0; count = 0; }

    //=====================================================================
    // Add the prototypes for the functions and inline functions here
    //=====================================================================







};
//=====================================================================
// Add the bodies for the functions here
//=====================================================================
```

13. (5 pts) The following code has many syntax and logical errors. Circle each error you find and describe what the error is. You may circle more than five errors. You will not be penalized for any incorrect responses. The first five correct responses will count towards the points for this question.

    NOTE: This question continues on to the next page. Be sure to check the next page for additional errors.

```
#include <iostream>
#include <stdio.h>
using namespace std;

class DivideByZero {}

class InvalidNumber : exception {
  private:
    int num;
  public:
    InvalidNumber() : exception() { num = 0; }
    InvalidNumber(int n) : exception() { num = n }
    string what() {
       sprintf(m, "%d is an invalid number.", num);
       return string(m);
    }
};

template
T divide(T num, T divisor) throw(InvalidNumber) {
  if(divisor == 0)
    throw DivideByZero;

  if(num < 0)
    throw InvalidNumber(num);

  if(divisor < 0)
    throw InvalidNumber(num);
}

int main() {
  int candy, people, res;

  cout << "How many pieces of candy are there? ";
  cin >> candy;
  cout << "How many people are there? ";
  cin >> people;

  try {
    res = divide(candy, people);
```

```
        if(res == 0)
          cout << "There is not enough candy for everyone!\n";
        else
          cout << "Give " << res << " pieces of candy to each person.\n";
      }
      catch(DivideByZero) {
        cout << "Error: Cannot divide by zero people!\n";
      }
      catch(...) {
        cout << "Error: Unknown exception.\n";
      }
      catch(InvalidNumber) {
        cout << "Error: " << e.what() << endl;
      }

      return 0;
    }
```