CS 1022C-001 - Spring 2014 - Exam 2
University of Cincinnati

There are 7 questions for a total of 115 points.
Your total score will be computed out of 100.

Name: _____

Instructions

- Please read through this entire exam very carefully before starting.
- This exam is closed book
  - You may use a single piece of standard 8.5inch x 11inch US letter paper. You may write whatever you like on all 6 sides of the paper, as long it contains your name somewhere. You must turn this paper in with your exam.
- All work must be written on the exam pages in order to be graded. Any scrap paper used, must be the scrap paper provided during the exam period.
- For programming questions: Please be accurate with your C++ syntax: this includes appropriate use of braces, semicolons, and the proper use of upper/lowercase letters.
- No electronic devices may be used during the exam: this includes (but is not limited to) calculators, phones, tablets, and computers.
- You have 55 minutes to complete the exam.

DON'T PANIC!

Question:	1	2	3	4	5	6	7	Total
Points:	10	12	10	36	12	15	20	115
Score:								

Multiple Choice and True/False

1. Circle the **best** response.

[2] (a) Which of these data types is the largest on a 32-bit architecture?

A. `int`
B. `float`
C. `int*`
D. `they are all the same size`

Solution: D - `they are all 32 bits on a 32 bit architecture`

[2] (b) If a method (`foo`) is declared `virtual` in class `A` and class `B` is a subclass of `A`, then the `foo` method is automatically virtual in class `B`.

A. true
B. false

Solution: A - true

[2] (c) The object on which a member function is applied is the implicit parameter named:

A. `this`
B. `self`
C. `that`
D. `object`

Solution: A - `this`

[2] (d) Changing the implementation of a method in a derived class is called

A. overloading
B. extending
C. overloading
D. overdoing

Solution: C - overloading

[2] (e) Using the arrow operator, `->`, to call a method like `obj->method()` is equivalent to:

A. `obj*method()`
B. `obj..method()`
C. `*(obj.method())`
D. `(*obj).method()`

Solution: D

Fill in the blank

2. Fill in the blank with the best answer

[4] (a) If the class `B` extends the class `A`, and an instance of `B` is created, the constructors are called in this order: _____ then _____.

Solution: A then B

[4] (b) If the class `B` extends the class `A`, and an instance of `B` is destroyed, the destructors are called in this order: _____ then _____.

Solution: B then A

[4] (c) The termination condition of a recursive function is called the _____.

Solution: base case

Short Answer

3. Provide a short answer to the following questions, using complete sentences.

[5] (a) What is encapsulation? Why is it useful?

Solution: Encapsulation is the hiding of data to ensure that access to it and modification of it

[5] (b) What happens if you forget to delete an object that you obtained from the heap? What happens if you delete it twice?

Solution: If you forget to delete - memory leak. If you delete it twice - double free / crash

Code Analysis

4. Use the code listed below for all parts of this question. In this scenario each

```
// includes omitted to save space
class Folder; // forward declaration of Folder so it compiles
class Document {
public:
  Document(string name, int id, Folder* folder)
      : name(name), idNumber(id), folder(folder) {}
  virtual void setFolder(Folder* folder) {
    this->folder = folder;
  }
  virtual void print() {
    cout << "Document id: " << idNumber << " name: " << name << endl;
  }
protected:
  string name;
  int idNumber;
  Folder* folder; // parent folder, can be NULL
};

class Folder : public Document {
public:
  Folder(string name, int id, Folder* parent) : Document(name, id, parent) {
  virtual void print() {
    cout << "Folder: id: " << idNumber << " name: " << name << endl;
    for (int i = 0; i < documents.size(); i++) {
      documents.at(i)->print();
    }
  }
  vector<Document*> getDocuments() {
    return documents;
  }
  void addDocument(Document* doc) {
    doc->setFolder(this);
    documents.push_back(doc);
  }
private:
  vector<Document*> documents;
};
```

4 (a) A variable of type `Document*` can be assigned instances of type

A. `Document*`
B. `Folder*`
C. none of the above
D. all of the above

Solution: D - all of the above

4 (b) A variable of type `Folder*` can be assigned instances of type

A. `Folder*`
B. `Document*`
C. none of the above
D. all of the above

Solution: A - `Folder*` only

6 (c) Does the following code compile using only the code provided above?

```
1 Document d("input.txt", 1, NULL);
2 Folder f("myDir", 2, NULL);
3 f = d;
```

A. `yes`
B. `no`

Solution: no - they cannot be assigned since not pointers.

6 (d) What is the output of this code, using the classes defined above. *This code does compile and run.*

```
1 Document d("input.txt", 1, NULL);
2 Folder f("myDir", 2, NULL);
3 vector<Document> v;
4 v.push_back(d);
5 v.push_back(f);
6 for (unsigned int i = 0; i < v.size(); i++) {
7   v[i].print();
8 }
```

Solution:

```
Document id: 1 name: input.txt
Document id: 2 name: myDir
```

(All or nothing.)

6 (e) What is the output of this code, using the classes defined above?

```
1  Document* d = new Document("input.txt", 1, NULL);
2  Folder* f = new Folder("myDir", 2, NULL);
3
4  vector<Document*> v;
5  v.push_back(d);
6  v.push_back(f);
7  for (unsigned int i = 0; i < v.size(); i++) {
8    v[i]->print();
9  }
```

Solution:

```
Document id: 1 name: input.txt
Folder: id: 2 name: myDir
```

(All or nothing.)

10 (f) What is the output of the following code, using the classes defined above?

```
1  Folder* root = new Folder("/", 1, NULL);
2  Folder* home = new Folder("home", 2, root);
3  root->addDocument(home);
4  Folder* user = new Folder("student", 3, home);
5  home->addDocument(user);
6  Document* resume = new Document("resume.txt", 4, user);
7  Document* prog = new Document("program.cpp", 5, user);
8  user->addDocument(resume);
9  user->addDocument(prog);
10
11 root->print();
```

Solution:

```
Folder: id: 1 name: /
Folder: id: 2 name: home
Folder: id: 3 name: student
Document id: 4 name: resume.txt
Document id: 5 name: program.cp
```

5. Answer the following questions using the code listing provided below.

```
1  long calculate( long i, long j ) {
2    if ( j == 1 ) {
3      return i;
4    } else {
5      return i * calculate( i, j - 1 );
6    }
7  }
8
9  long calculate( long i ) {
10   return calculate( i, i );
11 }
```

[3] (a) What is the result when `calculate(3)` is called?

Solution: 27

[3] (b) What is the result when `calculate(5)` is called?

Solution: 3125

[6] (c) Describe what this code calculates:

Solution: `calculate(i)` calculations i times itself i times.

Programming Questions

15 6. Write a recursive method to perform binary search on an array. The incoming array will already be sorted and the method should return -1 in the event that the target is not found in the array. Recall that binary search works by comparing the middle position to the `target` value and narrowing the search space if `target` is higher/lower than the value in the middle.

```
int binarySearch(int arr[], int size, int target) {
  return binarySearch(arr, target, 0, size - 1);
}

int binarySearch(int arr[], int target, int first, int last) {
  // BEGIN ANSWER
```

Solution:

```
  if (first > last) {
    return -1;
  }
  int mid = (last - first) / 2 + first;
  if (arr[mid] == target) {
    return mid;
  } else if (target < arr[mid]) {
    return binarySearch(arr, target, first, mid - 1);
  } else {
    return binarySearch(arr, target, mid + 1, last);
  }

  // END ANSWER
}
```

20 7. This is two recursive problems in one. Write the functions necessary to do a "substring reverse" inside a string. For example, given the string
`"this is super swish."`
and the "substring" to review of `"is"`, the result would be:
`"thsi si super swsih."`

The way this works, is the substring passed in, `"is"` in this case, is found in the string passed in. Each occurrence of the substring found is reversed inside the string.

Recommendation: write a recursive function to partially reverse a string based on a starting and ending index. Write a recursive function to look for substrings (and trigger the reverse function if found). Write a bootstrap function if necessary. Here is the sample main code, note that the string `str` is reversed in place (i.e. pass-by-reference).

```
string str = "this is super swish.";
string sub = "is";
subReverse(str, sub);
cout << "SubReversed: " << str << endl;
```

Solution:

```
void reverseStr(string& str, int s, int e) {
  if (s >= e) {
    return;
  }
  char tmp = str[e];
  str[e] = str[s];
  str[s] = tmp;
  reverseStr(str, s + 1, e - 1);
}
void subReverse(string &str, const string &sub, int start) {
  if (str.size() - start < sub.size()) {
    return;
  }
  bool reverse = true;
  for (int i = 0; i < sub.size(); i++) {
    if (str[start + i] != sub[i]) {
      reverse = false;
      break;
    }
  }
  if (reverse) {
    reverseStr(str, start, start + sub.size() - 1);
  }
  subReverse(str, sub, start + 1);
}
void subReverse(string &str, const string &sub) {
  subReverse(str, sub, 0); }
```

continued space for question 7.