Question 1. (20 points) A little C++ hacking. One of the summer interns has been working on a small C++ class to hold a list of integers using an array to store the data. Items are added at the end, and if the current capacity of the list is not large enough to hold an additional item, the array is replaced with a larger one and the data is copied to the new array.

Here's what's been implemented so far:

```
class List333 {
 public:
  // construct empty list with capacity for 3 items
  List333();
  // destructor
  ~List333();
  // apppend integer n to the end of list
  void Append(int n);

 private:
  // representation
  int * items;   // list of integers
  int capacity;  // amount of space in array items
  int size;      // list items are stored in items[0..size-1]

  // private local method
  // Ensure array items has at least n elements.  If not, replace
  // items with a larger array having n elements and copy the
  // list contents to it.
  void EnsureCapacity(int n);
};

// implementations (so far)

List333::List333() {
  items = new int[3];
  capacity = 3;
  size = 0;
}

void List333::Append(int n) {
  EnsureCapacity(size+1);
  items[size] = n;
  size++;
}
```

(continued on next page – you may remove this page from the exam if you wish)

Question 1. (cont.) As you can see, our intern did not finish implementing the destructor for this class or the private method EnsureCapacity. For this question, complete the implementation of those two methods below. If no code is needed, leave the appropriate space blank.

Hint: be sure there are no memory management bugs.

```
// Destructor
List333::~List333() {

   delete [] items;

}


// Private method:
// Ensure array items has at least n elements.
void List333::EnsureCapacity(int n) {
  if (n > capacity) {
    int *newItems = new int[n];
    for (int k = 0; k < size; k++) {
      newItems[k] = items[k];
    }
    delete [] items;
    items = newItems;
    capacity = n;
  }
}
```

Question 2. (20 points) Virtual madness. The following C++ program compiles and runs without errors. What output does it produce? (Write your answer at the bottom of the page)

```
#include <iostream>
using namespace std;

class B {
 public:
          void f() { cout << "B::f" << endl; }
  virtual void g() { f(); cout << "B::g" << endl; }
};

class D: public B {
 public:
          void f() { cout << "D::f" << endl; }
  virtual void h() { g(); f(); cout << "D::h" << endl; }
};

int main() {
  B *one = new B();
  D *duo = new D();
  B *two = duo;

  one->g();
  cout << "---" << endl;
  duo->h();
  cout << "---" << endl;
  two->g();

  return 0;
}
```

Output:

```
B::f
B::g
---
B::f
B::g
D::f
D::h
---
B::f
B::g
```

Question 3. (20 points) On the following page, implement a C++ program to keep track of the latest prices of a collection of stocks. The input to the program is read from `cin` and is a sequence of stock symbols followed by their current price. For example:

```
msft  31.10
appl  617.42
appl 636.34
mmm    93.75
msft  30.78
goog  672.87
```

After the program reads all of the input, it should write the last price of each stock and the corresponding symbol to `cout`. For this example input, the output should be:

```
appl 636.34
goog 672.87
mmm  93.75
msft 30.78
```

Each stock symbol and price should be written on a single line, but you are not required to list the stocks in any particular order.

Requirements: Your program must:

- Use a `map<string,double>` STL data structure to store the stock symbols and prices.
- Use an iterator to access the contents of the map in order to print out its contents.

The standard `cin>>` operation can be used to read items from input, skipping whitespace between them. The following loop, for example, will read strings from standard input, skipping whitespace, and print each string on a separate output line:

```
string s;
while (cin >> s) {
  cout << s << endl;
}
```

`cin>>` can also be used to read numbers as well as strings.

(write your answer on the next page – you may remove this page from the exam if you wish)

Question 3. (cont.) Write your stock price program below. Some useful `#include` directives as well as a `using` directive have been provided for convenience. You should add any additional libraries or code that you need. You probably will not need nearly as much space as is provided.

```
#include <string>
#include <iostream>
#include <map>


using namespace std;

// write your program here

int main() {
  string symbol;     // current stock symbol and price from input
  double price;
  map<string, double> stocks;  // latest prices for each stock

  // read stock symbols and prices and
  // update stored price for each stock
  while (cin >> symbol >> price) {
    stocks[symbol] = price;
  }

  // print stocks and last prices
  map<string, double>::iterator it;
  for (it = stocks.begin(); it != stocks.end(); ++it) {
    cout << it->first << " \t" << it->second << endl;
  }

  return 0;  // optional
}
```

Question 4. (18 points) "Oh what a tangled web we weave…" Our other summer intern was trying to program a simple "echo" web server, like the one used as an example in CSE 333. He found a bunch of information on the web, some of which might be useful, and some of which might not. Here is the intern's list of system calls and functions, including some incomplete information about their parameters:

```
1.int accept(socket, sockaddr, socklen_t)
2.int bind(socket, sockaddr, socklen_t)
3.int close(int)
4.int connect(socket, sockaddr, socklen_t)
5.int getaddrinfo(node, service, hints, res)
6.int listen(socket, backlog)
7.ssize_t read(int, buf, int)
8.int socket(family, type, protocol)
9.ssize_t write(int, buf, nbyte)
```

Our poor intern unfortunately has no idea how to use this collection of functions to create a server that will handle echo requests from a client. Please help him out by writing the function numbers from the above list that need to be executed by a server, and list them in the order that they need to be executed to handle a client. If a listed function is not needed, don't include its number below. If two functions are used in the same step, you should list both of their numbers in a single space below. If there are other functions not in the above list that might be used in a server, don't worry about them. You don't need to include them in your list.

Start here: 5 8 2 6 1 7/9 3 .

Question 5. (20 points) Fun with concurrency!! Consider the following C program:

```
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>

int x = 0;

void * f(void * ignore) {
  int a = x;
  x++;
  printf("a=%d ", a);
  printf("x=%d ", x);
  return NULL;
}

int main() {
  pthread_t t1, t2, t3;
  int res;
  res = pthread_create(&t1, NULL, f, NULL);
  res = pthread_create(&t2, NULL, f, NULL);
  pthread_join(t1, NULL);
  res = pthread_create(&t3, NULL, f, NULL);
  pthread_join(t2, NULL);
  pthread_join(t3, NULL);

  return 0;
}
```

For this question, assume that the operation `x++` executes as a single operation and cannot be interrupted by another thread while it is updating variable x.

For each of the following output sequences, circle "yes" if it could be produced by some possible execution of the above program and circle "no" if it could never happen under any circumstances. (Hint: at least one of these sequences is possible.)

(YES) NO `a=0 x=1 a=0 x=2 a=2 x=3`

YES (NO) `a=0 x=1 a=0 x=2 a=0 x=3`

(YES) NO `a=0 x=1 a=1 a=0 x=2 x=3`

YES (NO) `a=0 x=1 a=2 x=2 a=2 x=3`

Question 6. (`0x17>>3` points) The **BEST PROGRAMMING LANGUAGE** is

a) C
b) C++
c) C#
d) Java
e) Objective-C
f) Subjective-C
g) PHP
h) Javascript
i) Ruby
j) PERL
k) Python
l) Visual basic
m) x86 assembler
n) Bash
o) sed
p) make
q) Whitespace
r) Turing machine
s) SML
t) Haskell
u) Scheme/Lisp
v) Ada
w) Pascal
x) Matlab
y) Fortran
z) Cobol
aa) It depends
bb) None of the above. The correct answer is _____ .
cc) I really don't care. Just give me my free points, please.
dd) I really do care, but just give me my free points, please.