ECS 40 **Practice Final #2 Key**

1. (30 points, All or nothing) In p7, the first two programs had your derived class objects print out base class data that was declared private. Describe two techniques you used to achieve this. Your answer should be quite short. Remember, we are just trying to determine if you wrote your own code—be specific as possible.

   For one program, I used a public accessor method of the base class to retrieve the private data's value. In the other program, I called the print() method of the base class to print out the information for the derived class

2. (28 points) Write a recursive implementation of strcmp. The prototype of strcmp is: int strcmp(const char *s1, const char *s2). If s1 > s2 then strcmp returns a positive value. If s1 == s2 then strcmp returns 0. If s1 < s2 then strcmp returns a negative value. The header is provided.

Pts	
	`int strcmp(const char *s1, const char *s2) {`
16	`  if(*s1 != *s2 || !*s1 || !*s2)`
4	`    return *s1 - *s2;`
8	`  return strcmp(s1 + 1, s2 + 1);`
	`} // strcmp()`

3. (12 points) Write a UNIX command line that will list the names of all the people who handed in main.cpp. I am in the home directory of cs40a. Each team turns into a directory in handin/p7; e.g. handin/p7/davis, handin/p7/bulcher, handin/p7/wong. Names are located in the first two lines of each main.cpp, so just print those lines.

```
Pts:    2     1       1     1       2   2   1  1  1
      find handin/p7 –name main.cpp –exec head -2 {} \;
```

4. (40 points) Based on the following shell script, test.sh, answer the questions. An example call would be `test.sh a.out 300`.

Line #	
1	`#! /bin/bash`
2	`count=$2`
3	
4	`while [[ $count -gt 0 ]] ; do`
5	`  rm sk.temp >& /dev/null`
6	` ps	grep $1 > sk.temp`
7	`  if test `cat sk.temp |wc -w ` -eq 0  ; then`
8	`    exit 0`
9	`  fi`
10	`  sleep 10`
11	`  (( count = count - 10 ))`
12	`done`
13	
14	`pkill $1`

a) (3 points) What does line #1 do?
   Tells the shell which program should run the script.

For the rest of the parts of this question, assume that the command line was: `test.sh editor.out 50`

b) (3 points)What would line #2 do?
   Assigns count the value of the second parameter, 50.

c) (3 points) What does line #4 do?
   Stays in the loop as long as count is greater than zero.

d) (4 points) What does line 5 do? Be complete.
   Removes the file sk.temp, and sends any error messages to /dev/null (so they are not printed.)

e) (4 points) What does line 6 do?
Writes all processes whose names match the first parameter, editor.out, to the file sk.temp.

f) (4 points) What do lines 7 to 9 do?
If the file sk.temp is empty then the script terminates.

g) (3 points) What does line 10 do?
Suspends the script for 10 seconds.

h) (3 points) What does line 11 do?
Subtracts 10 from count.

i) (3 points) What does line 14 do?
Terminates all processes with the same name as the first parameter,editor.out.

j) (10 points) What does the whole script do in general?
Permits a program given as the first parameter,editor.out, to run for the number seconds indicated by the second parameter, 50. Every ten seconds, it checks to see the program has terminated every ten seconds. If the program has terminated then the script terminates. If the program does not complete in the indicated time, then the script terminates it.

5. (20 points)

a) (10 points) Assuming I have declared "queue<int > Q", and that the Front and Back indices are initially zero. Show the contents of the queue after the following operations. (Note: there may be more positions than needed.) Also provide Front and Back indices after all operations have been executed..

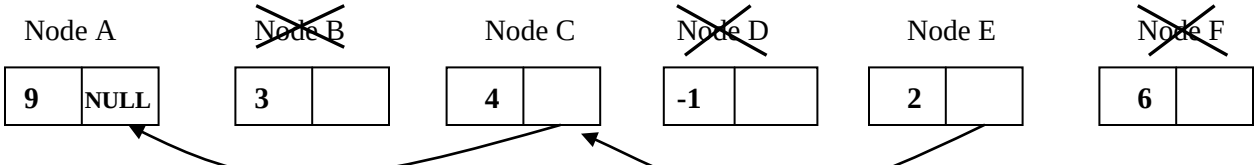
Q.push(7); Q.push(8); Q.pop(); Q.push(4); Q.push(2); Q.front(); Q.pop(); Q.push(-6); Q.back(); Q.push(3); Q.pop();

0	1	2	3	4	5	6	7	8
7	**8**	**4**	**2**	**-6**	**3**			

Front: **____3**_____ Back:___**6**_____ One point for each value in correct position, 2 points for Front & Back

b) (10 points) Assuming I have declared "stack<int, list<int> > st.) Show the state of the stack after the following operations. Do this by filling in the values and drawing the pointers (or entering NULL) for each list node that exists after the operations. Also provide the letter of the Node to which Top points after all operations are completed. Assume that the nodes are created in alphabetical order. If a node no longer exists, cross out its name, and make sure that no existing node points to it. (Note: there may be more nodes than needed.)

st.push(9); st.top(); st.push(3); st.pop(), st.push(4); st.top(), st.push(-1); st.pop(); st.push(2); st.push(6); st.pop()



Top : _____**E**___ **One point for each correct node, for B,D, and F crossed out, and Top of E.**

6. (171 points) For this section of the test, you will be writing parts of a program that reads information about products from two files, and then provides the information when provided a UPC (Uniform Product Code) by the user. A SKU is a unique string used by a company to identify a product. The class ProductInfo is derived from the class Product, and has an additional int UPC. SKUs.txt provides a mapping between SKUs and UPCs. products.csv provides information about a product based on its UPC with the format <UPC>, <Description>, <price>

a) (58 points) Provide a complete ProductInfo.h. Besides the required preprocessor directives, your file should contain all the appropriate pre-processor directives typical for large projects. The ProductInfo class is derived from the Product class, and has only an additional int UPC. It contains only a constructor, a print method, and two overloaded operators. Your file shold provide all of the implementation code for its member methods. You may wish to hold off writing parts of this until you have written some of the later functions. Remember to use const where appropriate.

Pts	
2	#ifndef productInfoH
2	#define productInfoH
2	#include <iostream>
2	#include "product.h"
3	using namespace std;
4	class ProductInfo : Product
	{
2	int UPC;
1	public:
10	ProductInfo(int U = 0):Product(string(""),0.0F),UPC(U) {}
2	friend istream& operator>> (istream &is, ProductInfo &info);
3	void print()const {
13	cout << UPC << " $" << setiosflags(ios::fixed) << setw(7) << setprecision(2) << price << " " << getName() << endl; }
12	bool operator< (const ProductInfo &rhs)const {return UPC < rhs.UPC;}
1	};
1	#endif

b) (27 points) Write the overloaded extraction operator for the ProductInfo class. The header is provided.

Pts	
	istream& operator>> (istream &is, ProductInfo &info){
	{
3	char s[256];
5	if(is.getline(s, 256))
	{
3	strtok(s, ",");
4	info.UPC = atoi(s);
5	info.setName(strtok(NULL, ","));
5	info.price = atof(strtok(NULL, ","));
	}
2	return is;
	}

c) (21 points) Write the readSKUs() function called from main. It reads SKU-UPC pairs from SKUs.txt into the map.

Pts	
4	void readSKUs(SKUtoUPCMap &SKUtoUPCmap)
	{
2	string SKU;
2	int UPC;
3	ifstream inf("SKUs.txt");
4	while(inf >> SKU >> UPC)
6	SKUtoUPCmap.insert(SKUtoUPCMap::value_type(SKU, UPC));
	// SKUtoUPCmap[SKU] = UPC; also OK
	} // readSKUs()

d) (37 points) Write the writeProductInfo() function called from main. It finds the UPC in map corresponding to SKU, and then uses the UPC to find ProductInfo in the set and calls the ProductInfo::print() method.

Pts	
12	`void writeProductInfo(const string &SKU, const SKUtoUPCMap &SKUtoUPCmap, const set<ProductInfo> &productSet)`
	`{`
6	`  SKUtoUPCMap::const_iterator itr = SKUtoUPCmap.find(SKU);`
5	`  if(itr == SKUtoUPCmap.end())`
2	`    cout << "Unknown SKU\n";`
1	`  else`
	`  {`
9	`    set<ProductInfo>::const_iterator itr2 =  productSet.find(ProductInfo(itr->second));`
2	`    itr2->print();`
	`  }`
	`} // writeProductInfo()`

e) Given the following definitions, write the insert function for LinkedList that keeps the list sorted by the UPC. The header is provided.

```
class ListNode {
  ProductInfo info;
  ListNode *next;
  ListNode(const ProductInfo &i, ListNode *n) : info(i), next(n){}
  friend class LinkedList;
};

class LinkedList{
  ListNode *head;
public:
  LinkedList():head(NULL){}
  void insert(const ProductInfo &info);
};
```

Pts	
	`void LinkedList::insert(const ProductInfo &info){`
4	`  ListNode *ptr, *prev = NULL;`
9	`  for(ptr = head; ptr && ptr->info < info; ptr = ptr->next)`
2	`    prev = ptr;`
2	`  if(prev)`
5	`    prev->next = new ListNode(info, ptr);`
1	`  else`
5	`    head = new ListNode(info, ptr);`
	`}`

```
typedef map<string, int> SKUtoUPCMap; // SKU's are strings, UPC's are ints

int main(int argc, char* argv[])
{
  SKUtoUPCMap SKUtoUPCmap;
  set<ProductInfo> productSet; // sorted by UPC
  ProductInfo info;
  string SKU;

  readSKUs(SKUtoUPCmap); // reads SKU-UPC pairs from SKUs.txt into the map.
  ifstream inf("products.csv"); // a comma separated value file.
  while(inf >> info)
    productSet.insert(info);

  do{
    cout << "SKU: ";
    cin >> SKU;
    writeProductInfo(SKU, SKUtoUPCmap, productSet);
      // finds UPC in map corresponding to SKU, and then uses the UPC
      // to find ProductInfo in set and calls the ProductInfo::print()
  } while (SKU != "0");

  return 0;
}


[davis@lect15 finalA]$ cat SKUs.txt
401-553 719980001
532-447 839220931
742-999 732228991
665-782 739816438
655-992 786539201
443-892 356271839
[davis@lect15 finalA]$


[davis@lect15 finalA]$ cat products.csv
719980001,Tony Chachere's Creole Seasoning,1.75
786539201,Dean Sniegowski P5 Solution,98.99
356271839,Kensington Track Ball,87.89
839220931,Bovine Online 7.0,4.5
732228991,Sony Walkman SRF-M35,24.99
739816438,ECS 40 Final Key,2999.99
[davis@lect15 finalA]$


[davis@lect15 finalA]$ products.out
SKU: 532-447
839220931 $   4.50 Bovine Online 7.0
SKU: 665-782
739816438 $2999.99 ECS 40 Final Key
SKU: 447-232
Unknown SKU
SKU: 443-892
356271839 $  87.89 Kensington Track Ball
SKU: 742-999
732228991 $  24.99 Sony Walkman SRF-M35
SKU: 0
Unknown SKU
[davis@lect15 finalA]$
```

```
class Product {
  string name;
protected:
  float price;
public:
  Product(string n, float p);
  const string& getName()const; // returns the name of Product
  float getPrice()const; // returns the price of the Product
  void setName(const string &n); //copies n to name
};
```