Test 2: CPS 08

Owen Astrachan

November 18, 1994

Name: __________________________

Honor code acknowledgement (signature) __________________________

<table>
<thead>
<tr>
<th>Problem 1</th>
<th>6 pts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 2</td>
<td>6 pts.</td>
</tr>
<tr>
<td>Problem 3</td>
<td>18 pts.</td>
</tr>
<tr>
<td>Problem 4</td>
<td>12 pts.</td>
</tr>
<tr>
<td>Problem 5</td>
<td>10 pts.</td>
</tr>
<tr>
<td>Problem 6</td>
<td>12 pts.</td>
</tr>
<tr>
<td>extra:</td>
<td>8 pts.</td>
</tr>
<tr>
<td>TOTAL:</td>
<td>64 pts.</td>
</tr>
</tbody>
</table>

PROBLEM 1:  (Vocabulary: 6 points)

For each of the words/phrases below, circle the definition that is the best description as it pertains in the context of computer science, programming, and C/C++.

1. **aliasing/shadowing**
   - (a) using the same identifier in a nested scope “hiding” the identifier in the outer scope.
   - (b) providing default values for parameters, e.g., as was done in the **ClockTime** class for 0 hours and 0 minutes and 0 seconds.
   - (c) using a reference parameter to avoid the overhead of making a copy of a value being passed.

2. **static**
   - (a) used as a type modifier — extends the lifetime of a local variable so that its value persists over multiple function calls
   - (b) used as a parameter modifier — prevents the value of the parameter from changing within a function
   - (c) used as a stream modifier — prevents the stream from “flowing”, i.e., stops values from being read

3. **ASCII**
   - (a) a library function in <ctype.h> for converting numbers to characters
   - (b) a standard coding for characters (each char has an underlying numerical value)
   - (c) something requested from an attendant at a BP service-station

PROBLEM 2:  (To Sir: With Name 6 points)

What is printed by the `cout << ...` statement below? Supply meaningful pre/postconditions for the function **Mystery**.

```cpp
void Mystery(String & name);

String nom = "Bjarne Stroustrup";
Mystery(nom);
cout << nom << endl;

void Mystery(String & name)
{
    String first, second;
    istrstream temp(name);
    first.ReadLine(temp, ' ');
    second.ReadLine(temp);
    name = second + "," + first;
}
```

2
PROBLEM 3:  (*I/O and Away 18 points*)

The program below prints six lines of output. Indicate each line of output in the space below to the right. The input to the program is shown below on the left.

```
#include <iostream.h>
#include "CPstring.h"

main()
{
    char ch1, ch2, ch3;
    String w1,w2,clear;

    cin >> ch1 >> ch2 >> ch3;
    cout << ch1 << ch2 << ch3 << endl;

    clear.getLine(cin);                       // read remaining chars on line
    cin.get(ch1);
    cin.get(ch2);
    cin.get(ch3);
    cout << ch1 << ch2 << ch3 << endl;

    clear.getLine(cin);                       // read remaining chars on line
    w1.getLine(cin,'a');
    cout << w1 << endl;

    w1.getLine(cin,'d');
    cout << w1 << endl;

    clear.getLine(cin);                       // read remaining chars on line
    cin >> w1 >> w2;
    cout << w1 << endl;
    cout << w2 << endl;
}
```

```
input
A big dog chased the bubble.
A big dog chased the bubble.
A big dog chased the bubble.
A big dog chased the bubble.
A big dog chased the bubble.
A big dog chased the bubble.
```

```
output
```
PROBLEM 4: (Window 12 points)

The number of meters a bicycle travels every 10 seconds is recorded in a file. As an example, the data below indicates that the bicycle travels $20 + 25 + 30 + 25 + 20 + 10 = 130$ meters in the first minute (60 seconds). The average speed is then 130 meters/min. After 20 more seconds (and 15 more meters) the average speed over the most recent minute will be 100 meters/min since the distance travelled in that time is $30 + 25 + 20 + 10 + 10 + 5 = 100$. You are to complete the program below so that it reads data from the file specified by the user and, for each reading after the first 6 (the first minute), prints the average speed in meters per minute for the most “recent” minute. For the data file above the output would be 130 120 100 75.

// appropriate include files here

main()
{
    String filename;
    ifstream input;
    int num;
    double total = 0.0;
    int readings = 0;

    cout << "enter file name: ";
    cin >> filename;
    input.open(filename);

    while(input >> num){

        if (readings > 6){
            cout << total << endl;
        }

    }
    input.close();
}
**Problem 5:**  
*Debugging 10 points*

**Part A** (4 points)

The function `atoi` in `<stdlib.h>` converts a string to a number. The function `itoa` below is intended to convert a number to a string. Although `itoa(6)` returns the string "6" and `itoa(666)` returns the string "666", the call `itoa(1234)` returns the string "4321". Fix `itoa` so that it works as intended by changing exactly one line in the function.

```c
String itoa(int num)
// precondition: num > 0
// postcondition: returns string form of num
{
    String val = "";
    while (num != 0){
        val = val + char('0' + num % 10);
        num /= 10;
    }
    return val;
}
```

**Part B** (6 points)

A function `NoDupes` designed to remove consecutive, duplicate elements from a sorted array is shown below. The picture below is intended to describe how the function works.

```
lastNonZero
??
```

The example below shows how the function should work on an array that initially contains 7 elements. *Add or alter at most three lines of code below so that NoDupes works as intended.*

```
before
| 2 | 2 | 3 | 4 | 4 | 4 | 5 |
-----
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

after
| 2 | 3 | 4 | 5 |
-----
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
```

```c
void NoDupes(int a[], int &numElts)
//precondition: a is sorted: a[0] <= a[1] <= ... <= a[numElts - 1]
//postcondition: duplicates in a removed, a still sorted
{
    int lastNonDup = 0;  // location of last non-duplicate
    int k;
    for(k=1; k < numElts; k++){
        if (a[k] != a[lastNonDup]){
            a[lastNonDup] = a[k];  // not a duplicate, store it
        }
    }
    numElts = ;  // potentially fewer elts, update
}
```
PROBLEM 6:  (Kroger 12 points)

part A (6 points)
A struct named Grocery can store information for items in a shopping cart. Given the definitions below, write the function CartWorth that determines how much all the items in cart cost together.

```cpp
enum Kind{heavy, squishy, frozen};

struct Grocery
{
    String brandName;
    double price;
    Kind type;
};

double CartWorth(const Grocery cart[], int numItems)
// precondition: numItems = # of grocery items in cart
// postcondition: returns total cost of all items in cart
{
}
```

part B: (6 points)
Assume information is stored in a file as shown below in the format

```
brand name : price universal product code
```

Froot Loops: 2.97:780079-116918
Skippy (Crunchy): 3.15:1201-3939
Haagan Das (Vanilla): 1.99:23913-19383
Wonder Bread: 0.99:91938-39393819
(a universal product code, or upc, is what’s scanned at the checkout as shown)

Write a function ReadCart to read information from such a file into an array of groceries. A stream bound to a file is passed to the function ReadCart. Assume that there’s a function StringToKind you may call that converts a upc to a Kind.

```cpp
Kind StringToKind(String upCode)
// precondition: upCode is a valid universal product code
// postcondition: returns a Kind : heavy, squishy, frozen
```
EXTRA CREDIT: (8 points)

Write a function `ReArrange` that re-arranges the items in a grocery cart so that all the heavy ones come before all the frozen ones which come before all the squishy ones:

```
heavy  frozen  squishy
```

You can assume that any useful `Swap` function exists. As a hint, keep the order shown and a "needs to be processed" part at the right end of the array. Initially the entire array needs to be processed. If the item being processed is frozen, it's in the right section — otherwise some swapping is needed.