

Test 1: CPS 006

Owen Astrachan and Susan Rodger

October 6, 1999

Name: _____

Login: _____

Honor code acknowledgment (signature) _____

	value	grade
Problem 1	12 pts.	
Problem 2	26 pts.	
Problem 3	24 pts.	
Problem 4	14 pts.	
TOTAL:	76 pts.	

This test has 10 pages, be sure your test has them all. Do NOT spend too much time on one question — remember that this class lasts 50 minutes.

In writing code you do not need to worry about specifying the proper `#include` header files. Assume that all the header files we've discussed are included in any code you write.

Class declarations for the classes `Dice` and `Date` are on the last page of the test. You can remove this page to make it easier to have a reference to these classes when needed.

PROBLEM 1 : (*Vocabulary*: 12 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. *Function Prototype*

- (a) The name and the parameter list of a function that together comprise a function's signature.
- (b) A boolean expression that should be true when the function is called if the function is to work as intended.
- (c) The header file `<iostream>` that is needed in programs that do input or output.

2. *pow*

- (a) The standard output stream.
- (b) The function in `<cmath>` that calculates the value of one number raised to the power of another number.
- (c) The string function that removes all characters from a string

3. *Loop Invariant*

- (a) The expression that updates variables in a loop test.
- (b) A boolean expression that is true each time a loop test is evaluated.
- (c) A programming guideline that recommends avoiding the `break` statement in a loop.

4. *constructor*

- (a) A program whose output is another program.
- (b) A member function that initializes an object.
- (c) A function in `<cmath>` that converts radian measure to degree measure.

PROBLEM 2 : (*Holidays*)

The problem below includes a function `NthDay` that will be the basis of several questions. The output of the program is shown before the program as is the first question. The line numbers shown in the program listing are not part of the program, but are included to make it possible to refer to specific parts of the program. Note that Labor day is always the first Monday in September and that Thanksgiving is the fourth Thursday in November.

```
prompt> days
enter a year 1999
labor day on September 6 1999
Thanksgiving on November 25 1999
```

Part A, 4 points

A user misspells a day of the week as shown below: note that "Monday" is not spelled correctly.

```
Date laborDay = NthDay(9, 1999, 1, "Modnay");
```

Explain in a few sentences what happens when `NthDay` is called with the misspelling.

```

0  #include <iostream>
1  using namespace std;
2  #include "date.h"
3
4  Date NthDay(int month, int year, int n, string day)
5  // pre: day = "Sunday" or "Monday" or "Tuesday" or ... "Saturday"
6  // post: returns a Date that is the n-th occurrence of day in
7  //       month/year
8  {
9      Date d(month,1,year);
10
11     while (d.DayName() != day)
12     {
13         d++;
14     }
15
16     // d is now the first "day" in the month
17
18     return d + (n-1)*7;
19 }
20
21 int main()
22 {
23     int year;
24     cout << "enter a year ";
25     cin >> year;
26
27     Date laborDay = NthDay(9,year,1,"Monday");
28     Date turkeyDay = NthDay(11,year,4,"Thursday");
29
30     cout << "labor day on " << laborDay << endl;
31     cout << "Thanksgiving on " << turkeyDay << endl;
32
33     return 0;
34 }

```

Part B, 6 points

Complete the calls of `NthDay` below so that they set variables as follows (for the year 1999):

variable	holiday	occurs on
<code>kingDay</code>	Martin Luther King Day	third Monday in January
<code>columbusDay</code>	Columbus Day	second Monday in October
<code>armedDay</code>	Armed Forces Day	third Saturday in May

```

Date kingDay = NthDay(    , 1999,    ,    );
Date columbusDay = NthDay(    , 1999,    ,    );
Date armedDay = NthDay(    , 1999,    ,    );

```

Part C, 4 points

Explain in a sentence or two why the value returned on line 18 from the function `NthDay` is correct.

Part D, 12 points

Write the function `WeekDays` that returns the number of weekdays (Monday–Friday, i.e., not a weekend) between and including the Dates specified by `first` and `last`. For example, the call below prints the number of weekdays between the day after Labor Day and the day before Thanksgiving.

```
cout << "# week days = "  
      << WeekDays(NthDay(9,year,1,"Monday") + 1,  
                  NthDay(11,year,4,"Thursday") - 1) << endl;
```

Complete `WeekDays` below:

```
int WeekDays(Date first, Date last)  
// pre: first <= last  
// post: returns number of weekdays ("Monday".."Friday") between  
//       and including first--last
```

PROBLEM 3 : (*Ricky Don't Lose That Number*)

In this problem, telephone numbers have area codes followed by seven digits, i.e., (919) 660-6595 is a telephone number, but 660-6595 isn't, because it doesn't include the area code.

There are many ways to write a telephone number, three ways are shown below:

919.660.6595 (919)660-6595 919 660 6595

Part A, 10 points

All currently assigned area codes in the U.S. have the following properties:

- They are three digits long
- The first digit cannot be a 0 or a 1
- The last two digits cannot be the same

Write the function `LegalAreaCode` below, here are some examples:

call	returned result	reason
<code>LegalAreaCode("109")</code>	false	first digit is a 1
<code>LegalAreaCode("919")</code>	true	
<code>LegalAreaCode("211")</code>	false	last two digits same
<code>LegalAreaCode("abc")</code>	false	not all digits
<code>LegalAreaCode("1234")</code>	false	not three digits long

To make it easier to write `LegalAreaCode` and other functions that use telephone numbers, the two helper functions below have been written, you can call them in functions/programs you write:

```
bool IsNumber(string s)
// post: returns true if s is a string of length 1, and
//       s is either "0","1",... , or "9"; returns false otherwise
{
    if (s.length() != 1) return false;
    return s == "0" || s == "1" || s == "2" || s == "3" || s == "4" ||
           s == "5" || s == "6" || s == "7" || s == "8" || s == "9";
}

string OnlyNumbers(string s)
// post: return a string with same characters as s, but in
//       which all non-numeric characters are removed
// example: s is "(919) 555-1212", returns: "9195551212"
{
    int k;
    int len = s.length();
    string only = "";
    for(k=0; k < len; k++)
    { if (IsNumber(s.substr(k,1)))
      { only += s.substr(k,1);
      }
    }
    return only;
}
```

Write `LegalAreaCode` below, you can call `IsNumber` and `OnlyNumbers`.

```
bool LegalAreaCode(string s)
// post: returns true if s is a legal area code, otherwise returns false
```

Part B, 8 points

Write a function that creates a number at random that's a valid area code. For example `RandomCode()` could return 323 or 509, but it cannot return 157 or 744 since those aren't legal area codes. All the legal area codes should be equally likely. Hint: adding one to an eight-sided dice roll yields a random number in the range $[2..9]$, subtracting one from a ten-sided dice roll yields a random number in the range $[0..9]$.

```
int RandomCode()
// post: returns an int that's a legal area code
```

Part C, 6 points

Write the function `SplitNumber` that returns both the area code and the four-digit number that comes at the end of a number, after the area code and the prefix.

For example

```
string t, area, number;
t = "919-684-1423";
SplitNumber(t, area, number);
```

would give `area` the value "919" and `number` the value "1423". The same values should be given to `area` and `number` when `t` is assigned "919.684.1423" or "(919)6841423".

```
void SplitNumber(string s, string & area, string & num)
// pre: s is a legal telephone number
// post: sets area to area code of s, and num to 4-digit number of s
//       if s isn't a legal telephone number, the values of
//       area and num can be anything (no error checking)
```

PROBLEM 4 : (*The Dot Com Problem*)

Part A, 6 points

The function `Suffix` below returns the suffix in a URL, for example it returns "edu" from the URL "http://www.cs.duke.edu" and "com" from the URL "google.com". Two assignment statements in the body of the loop of `Suffix` are incomplete, complete those two statements, and make no other changes to `Suffix`, so that it works as specified.

```
string Suffix(string url)
// post: returns the part of url that comes after the last dot
// example: for "www.yahoo.com" returns "com",
//           for "duke.edu" returns "edu"
//           for "foo" returns "" since there is no suffix
{
    int pos = url.find(".");

    if (pos == string::npos)
    {
        return "";           // there's no .suffix
    }

    // there is at least one "." in url if we reach here

    do
    {
        url =

        pos =

    } while (pos != string::npos);

    return url;
}
```

Part B, 8 points

You have been hired by a Vegetarian Pizza company to help automate its orders. Toppings for a pizza are specified in a single string, with each topping separated from others by a double-colon "::" as in the examples below.

```
pepperoni::onion::sausage
onion::green pepper::tomato
sausage::spinach
onion
```

The function `FirstTopping` returns the first topping in a string of toppings so that

```
string s = FirstTopping("onion::sausage::pepperoni");
string t = FirstTopping("tomato");
```

assigns to `s` the string "onion" and to `t` the string "tomato". You don't write `FirstTopping`, it's written and you can call it.

```

string FirstTopping(string top)
// pre: toppings in top are separated by "::"
// post: returns the first topping in top
{
    int pos = top.find "::");
    if (pos == string::npos) return top; // only one topping
    return top.substr(0,pos); // return the first topping
}

```

Write the function `NoMeat` whose header is given below. The function returns a string that's a list of toppings, with the same number of toppings as in parameter `top`, and with each topping separated from others by `::`, but in which each occurrence of `"sausage"` and `"pepperoni"` is replaced by `"tomato"`.

For example

```
string s = NoMeat("pepperoni::onion::sausage");
```

sets `s` to `"tomato::onion::tomato"`.

```

string NoMeat(string top)
// pre: toppings in top are separated by "::"
// post: string of toppings is returned with the same number
//       of toppings as in top, but in which all occurrences of
//       "pepperoni" or "sausage" in top are replaced by "tomato"

```