

VERSION B

CS 31
Winter 2016
Midterm Exam
February 4, 2016

Problem #	Possible Points	Actual Points
1	10	10
2	10	7
3	10	10
4	10	5
5	10	8
6	20	20
7	10	10
8	20	20
TOTAL	100	90

FIRST NAME: Maxwell

LAST NAME: Wang

STUDENT ID #: 504598149

SIGNATURE: 

CLOSED BOOK

ONE 8.5"x11" SHEET OF NOTES ALLOWED

NO ELECTRONIC DEVICES

GOOD LUCK ONE AND ALL!

1. [10 points in all]

[5 points] Which of these statements is true about the following program?
(Circle a, b, or c; if you circle b or c, briefly state the problem):

a. It is a well-formed C++ program with well-defined behavior if the user types BRUIN BEAR!.

b. It has no error that would prevent compilation, but when executed, if the user types BRUIN BEAR!, it has undefined behavior (e.g., it might crash or produce strange results). Briefly state what the problem is:

text[9] and text[1] are both the letter R, so they share an ASCII index number. When you subtract them, you end up with 0, and this will cause problems as you attempt to divide by 0.

c. It has at least one error that will prevent compilation from succeeding. Briefly state what the problem is:

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    cout << "Enter a kind of mascot: ";
    string text;
    getline(cin, text);
    if (text.size() >= 9 &&
        text[1]/(text[9] - text[1]) > 10)
        cout << "Done" << endl;
}
```

$\frac{12}{A-R} > 10$

+5

[5 points] Which of these statements is true about the following program?
(Circle a, b, or c; If you circle b or c, briefly state the problem):

- a. It is a well-formed C++ program with well-defined behavior. *Dis super risky though.*
- b. It has no error that would prevent compilation, but when executed, it has undefined behavior. Briefly state what the problem is:

- c. It has at least one error that will prevent compilation from succeeding. Briefly state what the problem is:

```
#include <iostream>
using namespace std;
int main()
{
    int a[5];
    int k;
    for (int n = 3; n > 0; n--)
    {
        if (n == 1)
            a[k] = 42;
        else if (n == 2)
            k = 4;
        else
            a[4] = 13;
    }
    cout << "Done" << endl;
}
```

13
3 4
a[4] = 13, n = 2
k = 4, n = 1
a[4] = 42.

2. [10 points] Circle each of the following program fragments that contain an infinite loop, if any.

Program Fragment 1

```
int n = 13;
while (n > 0 && n/3 != 1)
{
    cout << n << endl;
    n--;
    if (n % 2 != 0) // a reminder: remainder
        n -= 3;
}
cout << "Finish: n = " << n << endl;
```

n=12
n=11, n=8
n=7 n=4

$4/3 = 1$



13
12
8

Program Fragment 2

```
int n = 16;
int log = 0;
for (int i = 1; i < n; i = i * 2)
    log++;
cout << n << " " << log << endl;
```

n = 4

2 4 8 16



Program Fragment 3

```
int one = 1;
int two = 1;
int next( 0 );
cout << one << " " << two << " ";
do {
    next = one + two;
    one = two;
    two = next;
    cout << next << " ";
} while( next != 12 );
```

Prints Fibonacci sequence, but jumps over 12.



one = 1 two = 1 next = 0
 next = 2 one = 1 two = 2
 next = 3 one = 2 two = 3
 next = 5 one = 3 two = 5
 next = 8 one = 5 two = 8
 next = 13 one = 8 two = 13

1 1 2 3 5 8 13

13

3. [10 points]

1. Write a single statement that declares an array of 9 doubles. Name the array `array`.
2. Write a single statement that reads a value from `cin` and stores it in the last element of `array`.
3. Write a loop that sets every element of `array` that has an index that is odd to twice the value read from `cin` from Step 2 above (If the input from `cin` were 3, for example, then after this loop has completed, the last element of `array` would have the value 3, and `array [1]`, `array [3]`, `array [5]` and so on... would each have the value 6).

You may assume that:
`#include <iostream>`
`using namespace std;`
have appeared previously.

1. `double array [9] ;`

2. `cin >> array [8] ;`

3. `for (int i = 1 ; i < 8 ; i++) { //start at 1, since 0 is not odd`
`if ((i % 2) != 0) {`
`array [i] = array [8] * 2 ; }`
`}`

$1/2 = 0 R 1$
 $2/2 = 0 R 0$
 $3/2 = 1 R 0$
 $4/2 = 1 R 0$

4. [10 points] The following program fragment comes from the Energy Calculator assignment that was Programming Project 2. Lost Angels DWP has requested you make code changes to give some business customers 20% off their bill. This conservation rebate should apply only if they used less energy this month as compared to last month. Please review the code shown below:

```
cout << "Customer Name: ";
getline( cin, name );
cout << "Energy Used (in kilowatt hours): ";
cin >> amount;
cin.ignore( INT_MAX, '\n' );
cout << "Customer Type: ";
cin >> type;
cout << "Energy Used last billing period (in kilowatt hours): ";
cin >> lastmonth;

// a bunch of code goes here to verify the data entered...

if (type == RESIDENTIAL_TYPE)
{
    // calculate the residential bill charges
    cost = RESIDENTIAL_SERVICE_CHARGE;
    cost += firsttier * RESIDENTIAL_TIER1;
    cost += secondtier * RESIDENTIAL_TIER2;
    cost += thirddtier * RESIDENTIAL_TIER3;
}
else
{
    // calculate the business bill charges
    cost = BUSINESS_SERVICE_CHARGE;
    cost += firsttier * BUSINESS_TIER1;
    cost += secondtier * BUSINESS_TIER2;
    cost += thirddtier * BUSINESS_TIER3;
}
cout.precision( 2 );
cout.setf( ios::fixed );
cout.setf( ios::showpoint );
// conservation rebate
if (amount < lastmonth)
{
    cost = cost - cost * (1 / 5);
}
cout << "The bill for " << name << " is $" << cost << endl;
```

4. (continued)

[10 points] Unfortunately, the program does not work as intended. What small changes to the program will fix its problems?

Either clearly indicate the changes that need to be made below or rewrite a portion of the code.

```
cout << "Customer Name: ";
getline( cin, name );
cout << "Energy Used (in kilowatt hours): ";
cin >> amount;
cin.ignore( INT_MAX, '\n' );
cout << "Customer Type: ";
cin >> type;
cout << "Energy Used last billing period (in kilowatt hours): ";
cin >> lastmonth;

// a bunch of code goes here to verify the data entered...

if (type == RESIDENTIAL_TYPE)
{
    // calculate the residential bill charges
    cost = RESIDENTIAL_SERVICE_CHARGE;
    cost += firsttier * RESIDENTIAL_TIER1;
    cost += secondtier * RESIDENTIAL_TIER2;
    cost += thirddtier * RESIDENTIAL_TIER3;
}
else
{
    // calculate the business bill charges
    cost = BUSINESS_SERVICE_CHARGE;
    cost += firsttier * BUSINESS_TIER1;
    cost += secondtier * BUSINESS_TIER2;
    cost += thirddtier * BUSINESS_TIER3;
}
cout.precision( 2 );
cout.setf( ios::fixed );
cout.setf( ios::showpoint );
// conservation rebate
if (amount < lastmonth) // should be
{
    cost = cost - cost * (1 / 5);
}
cout << "The bill for " << name << " is $" << cost << endl;
```

// these cout statements should use endl's at the end.

*if ((type != RESIDENTIAL_TYPE) && (amount < lastmonth))
{ cost = cost * (4/5); }*

10/5.0

// we could also just stick the if statement for amount < lastmonth in the else statement, as indicated. Both ways should work (though not together).

5. [10 points] Convert this switch statement to code that produces exactly the same output that does not use a switch statement.
(Please read the code carefully!)

```
char letter;  
// a bunch of code goes here that gives letter a value..  
switch (letter)  
{  
    case 'H':  
        cout << "Hillary" << endl;  
    case 'D':  
        cout << "Donald" << endl;  
        break;  
    case 'B':  
    case 'M':  
        cout << "Bernie Martin" << endl;  
        break;  
    default:  
        cout << "Bruin!" << endl;  
        break;  
}
```

```
if (letter == 'H') {  
    cout << "Hillary" << endl;  
    cout << "Donald" << endl; // since there was no break.  
}  
else if (letter == 'B' || letter == 'M') {  
    cout << "Bernie Martin" << endl; }  
else { cout << "Bruin!" << endl; }
```

D?

-2

6 [20 points] The owners of a dog kennel are trying to keep track of dog drop-offs and dog pickups by their customers. With a sensor as customers enter and leave, they are receiving a string each day that says, for example, "dd Dd DD", where each d represents that a dog has been dropped off by its owner, and each D means that a dog has been picked up by its owner and spaces just mean nothing has changed. The order of the letters in the string is the order that dogs have come and gone from the kennel, so in that example string, first 2 dogs were dropped off, then eventually 1 got picked up and 1 more dog got dropped off, and then eventually 2 dogs got picked up, etc.

The owners of the dog kennel would like to know that their kennel is operating well within its approved size. On the next page, write a function to help them; here is its prototype:

```
bool dogKennel(string data, int kennelSize, int& maximum);
```

`data` is the string of dog drop offs and pickups.
`kennelSize` is the maximum number of dogs who can be safely accommodated in the kennel simultaneously.
`maximum` has no particular value when the function is called; the function sets it as indicated below.

The function returns `true` if the maximum number of dogs in the kennel simultaneously never exceeded the `kennelSize`, or `false` if that maximum did exceed the `kennelSize` at some point. Regardless of the return value, the function must set `maximum` to the maximum number of dogs who were in the dog kennel at any one time.

Notwithstanding the above paragraph, the function must also return `false` if the `data` string contains any characters other than `d` or `D` or space, if `kennelSize` is negative, or if during the analysis of the data, it appears that at some point there were a negative number of dogs in the kennel.

Here are some examples of how a main routine could test this function:

```
int max;
assert(dogKennel("dd DdD D", 10, max) && max == 2);

// In this example, the maximum number of dogs in the kennel at
// once was 8 which exceeded the kennelSize of 6
assert(!dogKennel("d D dd dd dd dd D", 6, max) && max == 8);

// In this example, 2 dogs were dropped off and then 10 dogs
// were picked up, leading to a negative number of dogs in the
// kennel at one point!
assert(!dogKennel("dd DDDDD DDDDD", 20, max) && max == 2);
```

6 (continued) Write your dogKennel function here. (You do not have to write a main routine or #include directives.)

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

```
bool dogKennel(string data, int kennelSize, int& maximum)
```

```
{ int size = data.size(); int tempmax = 0;
```

```
int dogNumber = 0;
```

```
bool nexceeded = true;
```

```
for (int i = 0, i < size; i++) {
```

```
if (data[i] == 'd') {
```

```
dogNumber++; }
```

```
else if (data[i] == 'D') {
```

```
dogNumber--; }
```

```
else if (data[i] != ' ') {
```

```
nexceeded = false; }
```

```
if (dogNumber < 0) {
```

```
nexceeded = false; }
```

```
if (dogNumber > kennelSize) {
```

```
nexceeded = false; }
```

```
if (dogNumber > tempMax) {
```

```
tempMax = dogNumber; } } //end of for loop
```

```
if (kennelSize < 0) { nexceeded = false; }
```

```
maximum = tempmax;
```

```
return nexceeded;
```

```
}
```

7. [10 points] What is the first digit of your UCLA student ID number? 5
 What output is produced by the following program if its input is the first digit of your UCLA student ID number?

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

int bruin(int one, int& two);
void bear(int& three);
```

data = 5 today = 242016
 data = 4 today = 242016
 today = 50
 today = 3
 data = 3

```
int main()
{
  int data;
  cin >> data; // enter the first digit of your student id
  int today = 242016;
  today = bruin(today, data); today = 50
  bear(today);
  bruin(today, data);
  bear(data);
}
```

data = 5 today = 242016
 data = 4 today = 50
 today = 3

```
int bruin(int one, int& two)
{
  cout << two << "-" << one << endl;
  two--;
  one = 50;
  cout << one << "-" << two << endl;
  return( one );
}
```

```
void bear(int& three)
{
  if (three > 100)
    three = 7;
  else if (three > 7)
    three = 3;
  else if (three > 4)
    three = 1;
  cout << three << endl;
}
```

data = 5 today = 242016.
 data = 4 today = 50.
 today = 3 , data = 3

5 - 242016 ✓
 50 - 4 ✓
 3 ✓
 4 - 3 ✓
 50 - 3 ✓
 3 ✓

8. [20 points] Suppose we have two string arrays. One holds the names of all the students who are enrolled in a class. The other holds the names of all the people who attended Tuesday's lecture for that class. (Not everyone attending a lecture is necessarily enrolled in the class.) We'd like to find out how many students who are enrolled in the class did not attend the lecture. You'll write a function named `missing` that will count how many elements in one array do not appear in another.

The function takes four parameters:

`a1`, an array of strings

`n1`, the number of elements in `a1`

`a2`, an array of strings

`n2`, the number of elements in `a2`

The function returns an `int` that is the number of elements of `a1` that do not appear somewhere in `a2`. When you write the function, you may assume that `n1` and `n2` are nonnegative and that no two elements of `a1` are the same. (Two or more elements of `a2`, however, may be the same.)

Here is an example of how the function can be used:

```
string enrolled[5] = {
    "farinaz", "xiaoxiao", "ilya", "thanh", "mark"
};
string attended[6] = {
    "mark", "bedros", "xiaoxiao", "thanh", "luis", "thanh"
};
int k = missing(enrolled, 5, attended, 6);
// k is 2: Two elements of enrolled are not in attended
//         (farinaz and ilya).
```

Write your function on the next page.

8. (continued) Write your missing function here. (You do not have to write a main routine or #include directives.)

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

```
int missing(string a1[ ], int n1, string a2[ ], int n2)
```

```
{
    int n = 0;
    bool found = false;
    for (int i = 0; i < n1; i++) {
        string s = a1[i];
        bool found = false;
        for (int j = 0; j < n2; j++) {
            if (a2[j] == s) {
                found = true;
            }
        }
        if (!found) {
            n++;
        }
    }
}
```

```
return n; }
```



n1
n2