

CMPS 150 – Test #2

Name: KEY

Section: _____

1. What is output of the following code?

```
sum = 0;
for (a = 7; a < 9; a++)
{
    k = a;
    while (k <= 10)
    {
        sum = sum + k;
        k++;
    }
}
cout << sum << endl;
```

61

2. What is the output of the following code? Assume the user enters 2 as the first integer, and 5 as the second.

```
cout << "Enter two integers: ";
cin >> num1 >> num2;

result = 1;
while (num2 > 0)
{
    result = result * num1;
    num2--;
}
cout << result << endl;
```

32

3. The following code is a while loop within a for loop. Re-write the code as a for loop within a while loop (that produces the same output as the original code).

```
for (row = 1; row <= 3; row++)
{
    cout << endl << '$';
    digit = 1;

    while (digit <= 14)
    {
        cout << '9';
        digit = digit + 2;
    }
}
```

```
row = 1;
while (row <= 3)
{
    cout << endl << '$';

    for (digit = 1; digit <= 14; digit = digit + 2)
        cout << '9';

    row++;
}
```

NOTE: I do NOT want to know the output of the code.

4. The distance traveled by a vehicle can be calculated as: $\text{distance} = \text{speed} * \text{time}$
Using the input statements below (that ask for the speed and hours of travel), write the code fragment to display a table of the number of miles traveled after each hour of the travel time. See sample run for an example.
NOTE: Output does NOT have to "line up neatly."

```
cout << "Enter speed (in mph): ";  
cin >> speed;
```

```
cout << "Enter hours traveled: ";  
cin >> hours;
```

```
for (i = 1; i <= hours; i++)  
{  
    distance = i * speed;  
  
    cout << i << "    " << distance << endl;  
}
```

sample run

```
Enter speed (in mph): 40  
Enter hours traveled: 3
```

Hour	Distance
1	40
2	80
3	120

OR

If they display hours in descending order,
take off only 1 point.

```
sum = 0;  
  
for (i = 1; i <= hours; i++)  
{  
    sum = sum + speed;  
  
    cout << i << "    " << sum << endl;  
}
```

Total Points: 6

for / while loop	
initialize to 1	1
continue if <= hours	1
increment by 1	1
computation of distance	1
output of hours	1
output of distance	1

5. Assume an employer agrees to pay an employee one (1) cent for his first day of work, two (2) cents for his second day of work, four (4) cents for his third day or work, That is, each day the employee's daily pay rate is doubled. Write the code fragment to determine the employee's daily rate at the end of his first month, i.e., after 30 days of work. (Remember, just write the code to do it – you don't actually calculate it by hand.)
HINT: This will be a loop that keeps track of a sum, BUT instead of using "the usual" counter variable in the sum statement (sum = sum + i), you will instead use the continuously changing daily pay rate.

```
sum = 0;
rate = 0.01;

for (i = 1; i <= 30; i++)
{
    sum = sum + rate;
    rate = rate * 2;
}
```

OR

```
sum = 0;
rate = 0.01;

for (i = 1; i <= 30; i++)
{
    sum = sum + rate;
    rate = rate * 2;
}
```

OR

```
// In this solution, the answer would be in number of pennies, as opposed
// to a dollar amount (e.g., 1024 pennies instead of $10.24)
```

```
sum = 0;
rate = 1;

for (i = 1; i <= 30; i++)
{
    sum = sum + rate;
    rate = rate * 2;
}
```

Total Points: 6

initialize rate to 1 cent	1
for / while loop	
initialize to 1	1
continue if <= 30	1
increment by 1	1
computation of rate	2

6. Write a code fragment to process a file of bank transactions. There are an unknown number of transactions in the file, and each transaction is either a deposit or a withdrawal. The file (see sample) is formatted such that each line contains a transaction type (either a 'D' or a 'W') and a transaction amount. Transactions of type 'D' indicate a deposit, and the transaction amount must be added to the current balance. Transactions of type 'W' indicate a withdrawal, and the transaction amount must be subtracted from the balance. The original balance is input by the user. You may assume the file is open and the `ifstream` variable is `inFile`.

When done processing the file, print the final balance, as well as the total number of deposits, and the total dollar amount of all deposits. You may assume all deposits and withdrawals are positive values (this should help you decide what to initialize your maximum withdrawal to). The following input statement get the beginning balance for you.

```
cout << "Enter the beginning balance: ";
cin >> balance;
```

sample input file

```
D 249.52
W 27.68
W 74.35
W 31.99
D 286.27
...
...
```

```
countDeposits = 0;
```

```
sumDeposits = 0;
```

```
inFile >> type >> amount;
```

```
while(!inFile.eof())
{
    if (type == 'D')
    {
        balance = balance + amount;

        countDeposits++;

        sumDeposits = sumDeposits + amount;
    }
    else
    {
        balance = balance - amount;
    }

    inFile >> type >> amount;
}
```

```
cout << "Balance: " << balance << endl;
```

```
cout << "Total Number of Deposits: " << countDeposits << endl;
```

```
cout << "Total Amount of Deposits: " << sumDeposits << endl;
```

Total Points: 21

initialize countDeposits to 0	1
initialize sumDeposits to 0	1
priming read	2
while loop based on !eof	1
if statement for type 'D' (if missing ', penalize 1 pt.)	2
statements inside if type 'D'	
update balance	1
update countDeposits	1
update sumDeposits	1
else statement (or another if type 'W')	1
statements inside else (or type 'W')	
update balance	1
repeat priming read	1
output statements (1 pt. each) (no "pretty" stuff required)	3