MIDTERM EXAM

EMPLID □□□□□□□□

CSCI 135

NAME: FIRST LAST □

1. (12%) Suppose your program has the following declarations to represent information about a student:

```
string major ; // possibly empty
float gpa ;
bool female ;  // true if female , false if male
```

Write C++ logical conditions corresponding to each of the following sets. Your answers should be as compact
as possible and cover all cases.

(a) Female computer science majors with GPAs between 3.5 and 3.9.

```
female && major == "computer science" && gpa > 3.5 && gpa < 3.9
```

(b) Male students, whose major starts with the letter 'e' (economics, english, etc), and whose GPA is 2.0 or lower.

```
!female && major[0] == 'e' && gpa < 2.0
```

(c) All students, whose major ends in the letter 's' (mathematics, physics, etc), and whos GPA is a perfect 4.0.

```
major[major.length() - 1] == 's' && gpa == 4.0
```

2. (10%) Write a C++ function that calculates: $\sqrt{\frac{137(x-y)}{z^{n-1}}}$

```
#include <cmath>
double foo(double x, double y, double z, double n)
{
  return sqrt(137 * (x - y) / pow(z, n - 1));
}
```

3. (18%) Consider the following program fragment:

```
int enigma (int a, int & b);
```

(!!! this is a dashed box

```
int main() {
    int x = 0; // "SPECIAL LINE"
    cout << x++;
    cout << ++x << endl;
    for (int k = 1; k < 3; k++)
        cout << enigma(k , x);
    return 0 ;
}

int enigma(int a, int & b) {
    static int c = 0;
    c = a++;
    b += 2;
    return c * b;
}
```

(a) What does the program output?

0 2

4 12

(b) Circle all actual arguments in the program.

(c) Underline all formal parameters in the program.

(d) Draw a dashed box around all prototypes in the program.

(e) Draw a solid box around the scope of the variable declared on SPECIAL LINE?

(f) What is the value of variable `c` at the end of program execution - just before the `main()` function returns?

2

4. (15%) Write a function: `void average_word_lenght(string & sentence, float & result)` that calculates the average length of all words in the string `sentence`.

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

void average_word_lenght(string & sentence, float & result) {
    float sum = 0;
    int count = 1;
    for (int i = 0; i < sentence.length(); i++)
    {
        if (sentence[i] == ' ') {
            count++;
        }
        else {
            sum++;
        }
    }
    result = sum / count;
}

int main()
{
    string sentence = "";
    float result;
    average_word_lenght(sentence, result);
    cout << result << endl;
    return 0;
}
```

5. (15%) Write a program that asks user for a positive integer side length. If they enter an illegal value, they must be prompted to enter a good one until they do. It then displays, using asterisks, a filled diamond of the given side length. For example, if the side length is 4, the program should display:

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

```
int main()
{
   cout << "Enter the length of the diamond side: ";
   int side;
   cin >> side;
   // Calculate the max width of the diamond
   int max_width = (side * 2) - 1;
   // Calculate half of that to place spaces
   int half_width = (max_width) / 2;
   int dots = 1;
   // Print top half of diamond
   for (int i = 0; i < side; i++)
   {
      // Print spaces to line up triangle
      for (int j = 0; j < half_width; j++)
      {
         cout << " ";
      }
      // Print dots
      for (int j = 0; j < dots; j++)
      {
         cout <<"*";
      }
      cout << endl;
      dots = dots + 2;
      half_width--;
   }
   // Reset variables for bottom half
   dots = max_width - 2;
   half_width = 1;
   // Print bottom half of diamond
   for (int i = 0; i < side; i++)
   {
      // Print spaces to line up triangle
      for (int j = 0; j < half_width; j++)
      {
         cout << " ";
      }
      // Print dots
      for (int j = 0; j < dots; j++)
      {
         cout <<"*";
      }
      cout << endl;
      dots = dots - 2;
      half_width++;
   }
   return 0;
}
```

Variable and Constant Definitions

```
 Type    Name        Initial value
int cans_per_pack = 6;

const double CAN_VOLUME = 0.335;
```

Mathematical Operations

#include <cmath>

`pow(x, y)`	Raising to a power	x^y
`sqrt(x)`	Square root	$\sqrt{x}$
`log10(x)`	Decimal log	$\log_{10}(x)$
`abs(x)`	Absolute value	$\|x\|$

`sin(x)`
`cos(x)`
`tan(x)`
	Sine, cosine, tangent of x (x in radians)	

Selected Operators and Their Precedence

(See Appendix B for the complete list.)

`[]`	Array element access
`++ -- !`	Increment, decrement, Boolean *not*
`* / %`	Multiplication, division, remainder
`+ -`	Addition, subtraction
`< <= > >=`	Comparisons
`== !=`	Equal, not equal
`&&`	Boolean *and*
`||`	Boolean *or*
`=`	Assignment

Loop Statements

```
                  Condition
while (balance < TARGET)
{
   year++;                                       Executed
   balance = balance * (1 + rate / 100);         while condition
}                                                is true

     Initialization  Condition  Update
for (int i = 0; i < 10; i++)
{
   cout << i << endl;
}

                 Loop body executed
do                  at least once
{
   cout << "Enter a positive integer: ";
   cin >> input;
}
while (input <= 0);
```

Conditional Statement

```
               Condition
if (floor >= 13)
{                                 Executed when
   actual_floor = floor - 1;      condition is true
}
else if (floor >= 0)              Second condition (optional)
{
   actual_floor = floor;
}
else
{                                      Executed when all
   cout << "Floor negative" << endl;   conditions are false
}                                      (optional)
```

String Operations

```
#include <string>
string s = "Hello";
int n = s.length(); // 5
string t = s.substr(1, 3); // "ell"
string c = s.substr(2, 1); // "l"
char ch = s[2]; // 'l'
for (int i = 0; i < s.length(); i++)
{
   string c = s.substr(i, 1);
   or char ch = s[i];
   Process c or ch
}
```

Function Definitions

```
  Return type         Parameter type and name
double cube_volume(double side_length)
{
   double vol = side_length * side_length * side_length;
   return vol;
}                   Exits function and returns result.
Reference parameter
void deposit(double& balance, double amount)
{
   balance = balance + amount;
}                 Modifies supplied argument
```

Arrays

```
 Element type   Length
int numbers[5];
int squares[] = { 0, 1, 4, 9, 16 };
int magic_square[4][4] =
{
   { 16, 3, 2, 13 },
   { 5, 10, 11, 8 },
   { 9, 6, 7, 12 },
   { 4, 15, 14, 1 }
};

for (int i = 0; i < size; i++)
{
   Process numbers[i]
}
```