MIDTERM EXAM

CSCI 135

EMPLID | | | | | | | | |

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.

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

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

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

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

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

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?

(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?

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`.

5. (15%) Write a function: `bool equals(char a[], int a_size, char b[], int b_size)` that checks whether two char arrays have the same characters in the same order.

6. (15%) Write a function: `void bar_chart(int value_array[], int size)`
that displays a bar chart of the values in `value_array`, using asterisks, like this:

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

Your function must first check that all values in `value_array` are positive and no larger than 40. If a value falls outside of this range, no line should be printed for that value (not even blank one).

7. (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:

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

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 $\lvert x \rvert$
`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
                   at least once
do
{
   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

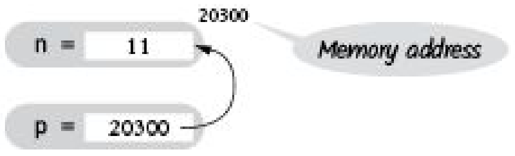
```
 Element type   Length
int numbers[5];
int squares[] = { 0, 1, 4, 9, 16 };
int magic_square[4][4] =
{
   { 16, 3, 2, 13 },
   { 5, 10, 11, 8 },
   { 9, 6, 7, 12 },
   { 4, 15, 14, 1 }
};

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

Vectors

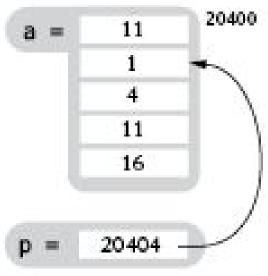
```
#include <vector>
                    Element type       Initial values (C++ 11)
vector<int> values = { 0, 1, 4, 9, 16 };
                                 Initially empty
vector<string> names;
                                    Add elements to the end
names.push_back("Ann");
names.push_back("Cindy"); // names.size() is now 2

names.pop_back(); // Removes last element

names[0] = "Beth"; // Use [] for element access
```

Pointers

```
int n = 10;
int* p = &n; // p set to address of n
*p = 11; // n is now 11
```



```
int a[5] = { 0, 1, 4, 9, 16 };
p = a; // p points to start of a
*p = 11; // a[0] is now 11
p++; // p points to a[1]
p[2] = 11; // a[3] is now 11
```



Input and Output

```
#include <iostream>
cin >> x; // x can be int, double, string
cout << x;

while (cin >> x) { Process x }
if (cin.fail()) // Previous input failed


#include <fstream>
string filename = ...;
ifstream in(filename);
ofstream out("output.txt");

string line; getline(in, line);
char ch; in.get(ch);
```

Range-based for Loop

```
                 An array, vector, or other container (C++ 11)
for (int v : values)
{
   cout << v << endl;
}
```

Output Manipulators

`#include <iomanip>`

`endl`	Output new line
`fixed`	Fixed format for floating-point
`setprecision(`*n*`)`	Number of digits after decimal point for fixed format
`setw(`*n*`)`	Field width for the next item
`left`	Left alignment (use for strings)
`right`	Right alignment (default)
`setfill(`*ch*`)`	Fill character (default: space)

Class Definition

```
class BankAccount
{
public:
   BankAccount(double amount);  — Constructor declaration
   void deposit(double amount); — Member function declaration
   double get_balance() const;  — Accessor member function
   ...
private:                   Data member
   double balance;
};

void BankAccount::deposit(double amount)  ┐
{                                         ├ Member function
   balance = balance + amount;            │ definition
}                                         ┘
```

Inheritance

```
                        Derived class       Base class
class CheckingAccount : public BankAccount
{
public:                                Member function
   void deposit(double amount);        overrides base class
private:
   int transactions;    — Added data member
};                        in derived class

void CheckingAccount::deposit(double amount)
{
   BankAccount::deposit(amount);  — Calls base class
   transactions++;                  member function
}
```