Midterm of CS 135, v2
Mar 17, 2025

Name:	
EmpID:	

1 (30 points) Answer the following questions.

(1) Given `int arr[] = {3, 7, 1, 2}`, what is arr[2]?

(2) Define function header for function **order**, for two double numbers a and b, if a is larger than b, swap them. Return type is void.

(3) Write a statement to generate a random **floating point number** in [0, 3]. No need to include libraries. Hint: you may use rand function, which returns a random integer in `[0, RAND_MAX]`.

(4) Given `string greeting = "Glad to meet you";` What is the value for greeting.substr(3, 6)?

(5) What is the value of `2 + 76 % 10`?

(6) What is the value of foo(5)?

```
int foo(int n) {
    int sum = 0;
    for (int i = 2; i <= n; i += 2)
        sum += i;

    return sum;
}
```

(7) Suppose double variables a and b are properly declared and initialized. Declare a double variable c and initialize it to be $\frac{\sqrt{a}}{2+b}$. You may use sqrt function, see cheat sheet.

(8) What is the output of the following code?

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

int main() {
    int count = 0;
    for (int i = 8; i >= 0; i -= 2)
        count++;

    cout << count << endl;
    return 0;
}
```

(9) What is the output of the following code?

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

void foo(int arr[], int size);
int main() {
    int arr[] = {1, 5, 3};
    int size = sizeof(arr) / sizeof(arr[0]);
    foo(arr, size);

    for (int i = 0; i < size; i++)
        cout << arr[i] << " ";

    return 0;
}

void foo(int arr[], int size) {
    for (int j = size-1; j >= 1; j--)
        for (int i = 0; i < j; i++)
            if (arr[i] < arr[i+1])
               swap(arr[i], arr[i+1]);
}
```

(10) Write a condition to represent that integer i is less than 0 **or** i is odd. An integer is odd if it cannot be divided by 2.

2 (20 points) Answer the following questions.

(1) What is the output of the following code?

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

void show(int size);
int main() {
    show(3);
    return 0;
}

void show(int size) {
    for (int row = 0; row < size; row++) {
        for (int col = 0; col < size; col++)
            if (row + col == size-1)
               cout << "-";
            else cout << "*";

        cout << endl;
    }
}
```

(2) Define function `no_upper`, for a string, return true if **none** of its elements is a uppercase letter, that is, one from 'A' to 'Z', otherwise, return false. For example, `no_upper("abc")` returns true and `no_upper("Ab1")` returns false. **No need to include libraries or define main function.**

Hint: you may use `int isupper(int ch)` to test whether a character is uppercase letter 'A' - 'Z' or not.

3 (50 points) Programming exercises

(1) Define function called `percentage`, for an array of ints, its size, and a target, return the percentage of integers that is **less than or equal to** the target to the size of the array.

For example, if the array has elements 3, 2, -1, and target is -1, then there is 1 integer out of 3 elements that is less than or equal to the target, so the percentage is 33.3333%. The return is 33.3333, a floating point number.

In main function, declare array `arr` with values 3, 2, -1. Call the above function on `arr` with appropriate size and target -1. Print out the return. **Just write the statements in main function, no need to include libraries**.

(2) Write code in main to enter a series of integers until -1 is entered. Find out the number of integers that can be divided by both 2 and 5 at the same time, the number of integers that can be divided by 2 only, and the number of integers that can be divided by 5 only.

Integer num is divided by 5 means the remainder of num divided by 5 is 0.

```
Enter an int (-1 to stop): 5
Enter an int (-1 to stop): 2
Enter an int (-1 to stop): 10
Enter an int (-1 to stop): 7
Enter an int (-1 to stop): 6
Enter an int (-1 to stop): 30
Enter an int (-1 to stop): -1
number can be divided by 2 and 5: 2
number can be divided by 2 only: 2
number can be divided by 5 only: 1
```

Variable and Constant Definitions

```
int cans_per_pack = 6;

const double CAN_VOLUME = 0.335;
```

Type, *Name*, *Initial value*

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++;
   balance = balance * (1 + rate / 100);
}
```

Executed 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)
{
   actual_floor = floor - 1;
}
else if (floor >= 0)
{
   actual_floor = floor;
}
else
{
   cout << "Floor negative" << endl;
}
```

Executed when condition is true

Second condition (optional)

Executed when all 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>
vector<int> values = { 0, 1, 4, 9, 16 };

vector<string> names;

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
```

Element type

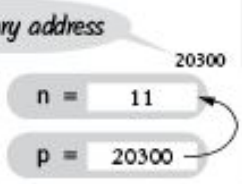
Initial values (C++ 11)

Initially empty

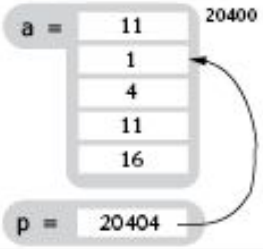
Add elements to the end

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);
```

Static Variables

```
void increment_print() {
   static int s_value = 0; //static duration
   s_value++;
   cout << s_value << '\n';
} //s_value is not destroyed, but goes out of scope
int main() {
   increment_print(); //1
   increment_print(); //2
}
```

Static Data Members

```
class Item {
private:
   int m_id;
   static int s_id_counter;
public:
   Item() {
      m_id = s_id_counter++;
   }
   int get_id() const {
      return m_id;
   }
};
int Item::s_id_counter = 1;
int main() { //
   Item first;
   Item second;
   cout << first.get_id();  //1
   cout << second.get_id();//2
}
```

Range-based for Loop

```
for (int v : values)
{
   cout << v << endl;
}
```

An array, vector, or other container (C++ 11)

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)

Enumerations, Switch Statement

```
enum Color { RED, GREEN, BLUE };
Color my_color = RED;

switch (my_color) {
   case RED :
      cout << "red";  break;
   case GREEN:
      cout << "green"; break;
   case BLUE :
      cout << "blue"; break;
}
```

Class Definition

```
class BankAccount
{
public:
   BankAccount(double amount);
   void deposit(double amount);
   double get_balance() const;
   ...
private:
   double balance;
};

void BankAccount::deposit(double amount)
{
   balance = balance + amount;
}
```

Constructor declaration

Member function declaration

Accessor member function

Data member

Member function definition

Inheritance

```
class CheckingAccount : public BankAccount
{
public:
   void deposit(double amount);
private:
   int transactions;
};

void CheckingAccount::deposit(double amount)
{
   BankAccount::deposit(amount);
   transactions++;
}
```

Derived class

Base class

Member function overrides base class

Added data member in derived class

Calls base class member function