Midterm of CS 135, v2
October 16, 2023

Name:	
EmpID:	

1 (30 points) Answer the following questions.

(1) Given `int arr[] = {-1, 2, 6, 3}`, what is arr[1]?

Answer: arr[1] is 2.

(2) Declare function **foo** whose first formal parameter is an int variable named **i** (pass by reference) and second formal parameter is a double variable named **d** (pass by value) and the return is a string. You just need to write the function header, no implementation is needed.

Answer: `string foo(int& i, double d);`

(3) What does `rand() % 2 + 5` produce?

Answer: `rand() % 2` generates a random integer 0, 1, so `rand() % 2 + 5` generates a random integer 5, 6

(4) Given `string greeting = "Wonderful";` What is the value for greeting.substr(1, 5)?

Answer: onder

(5) Suppose a runnable file is called run, write command to run it and redirect input from numbers.txt.

Answer: `run < numbers.txt`

(6) What is the value of 1 + 3 / 2 * 2?

Answer: 3

(7) Suppose n is 1275, what the value of digit after the following two statements?

```
n /= 10;
digit = n % 10;
```

Answer: 7

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

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

int main() {
    int sum = 0;
    for (int i = 6; i > 0; i -= 3)
        sum += i;

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

Answer: 9

(9) What is output for the following code?

```
    int a = 6;
    int* p = &a;
    *p /= 3;
    cout << a << endl;
```

Answer: 2

(10) What is the output for the following code?

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

void foo(int& a, int& b);

int main() {
    int i = 2;
    int j = 0;

    foo(i, j);

    cout << "i = " << i << ", j = " << j << endl;
    return 0;
}

void foo(int& a, int& b) {
    if (a > b) {
       a++;
       b--;
    }
}
```

Answer: i = 3, j = -1

2 (15 points) Answer the following questions.

(1) Write code to calculate $b^c - \sqrt{2a}$, and put the result to variable c. Assume that a, b, and c are properly defined and initialized double number. Hints: you may use pow and sqrt function.

Answer: c = pow(b, c) - sqrt(2 * a);

(2) Write code to

- enter a file name from console,
- declare and instantiate an ofstream object, call it **fout**, to open that file to write.
- NO need to check whether the file can be opened correctly or not. Also no need to write statements to write to the file or close it.

Answer:

```
cout << "Enter a file name: ";
string fileName;
cin >> fileName;
ofstream fout(fileName);
```

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

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

int main() {
    for (int row = 4; row >= 1; row--) {
        for (int col = 0; col < row; col++)
            cout << "?";

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

Answer:

```
????
???
??
?
```

3 (30 points) Programming exercises

(1) Write code in main to read a string (may contain spaces) from console, find out and print out the number of letter `'e'` in it. For example, suppose we enter "apple orange egg", then print out 3.

Here is a sample input/output:

```
Enter a string: apple orange egg
appearances of e: 3
```

Answer:

```cpp
//count number of appearances of letter 'a' in it.
//Sample input/output:
//Enter a string apple orange egg
//appearances of e: 3

#include <iostream>
using namespace std;

int main() {
    cout << "Enter a string: ";
    string line;

    getline(cin, line);

    int count = 0;
    for (int i = 0; i < line.size(); i++)
        if (line[i] == 'e')
           count++;

    cout << "appearances of e: " << count << endl;

    return 0;
}
```

(2) Define a function, for a given array of integers and its size, find out whether all its elements are a multiple of 3 or not.

For example, call the above function on array with values 1, 2, 3, the return is false. Call the above function on array with values 3, 6, 9, 0, the return is true.

Answer:

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

bool all_multiple_3(int arr[], int size);

int main() {
    int arr[] = {1, 2, 3};
    int size = sizeof(arr) / sizeof(arr[0]);
    cout << boolalpha << all_multiple_3(arr, size) << endl; //false
    //boolalpha print true for 1, false for 0.

    int arr2[] = {3, 6, 9, 0};
    int size2 = sizeof(arr2) / sizeof(arr2[0]);
    cout << boolalpha << all_multiple_3(arr2, size2) << endl; //even

    return 0;
}

bool all_multiple_3(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        if (arr[i] % 3 != 0)
           return false;
    }

    return true;
}
```

4 (15 points) Write codes

(12 points) Define a function named **productMaxMin**, for a given array of integers and its size, return the product of its maximum and minimum elements.

Answer:

```cpp
//Define a function to find out the product of
//the maximum and minimum elements of an array.
//sample output:
//product of max and min elements: -6
#include <iostream>
using namespace std;

int productMaxMin(int* arr, int size);

int main() {
    int arr[] = {-1, -2, 0, 3};
    int size = sizeof(arr) / sizeof(arr[0]);

    cout << "product of max and min element: "
         << productMaxMin(arr, size) << endl;
    return 0;
}

int productMaxMin(int* arr, int size) {
    int min = arr[0];
    int max = arr[0];

    for (int i = 1; i < size; i++)
        if (arr[i] < min)
           min = arr[i];
        else if (arr[i] > max)
                max = arr[i];

    return max * min;
}
```

(3 points) In main function, declare and initialize array with values -1, -2, 0, 3. Call the above function to print out its return.

No need to include library, using namespace statement, or write main function header. Just write the code in main function.

5 (10 points) Write codes

Define a function, for a given string, return a string with only digital letters in the original string.

For example, call the above function on "hello123, h1o2w a3r4e y5ou?", the return is "12312345".

You may use `int isdigit ( int c );` to test whether character is a decimal digit or not.

Answer:

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

string only_digit(string str);

int main() {
    cout << only_digits("hello123, h1o2w a3r4e y5ou?") << endl;
    return 0;
}

string only_digits(string str) {
    string result = "";
    for (int i = 0; i < str.size(); i++) {
        if (str[i] >= '0' && str[i] <= '9') //or if (isdigit(str[i])
          result += str[i];
    }

    return result;
}
```

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]
}
```

Vectors

Element type *Initial values (C++ 11)*

```
#include<vector>
vector<int> values = { 0, 1, 4, 9, 16 };
```

Initially empty

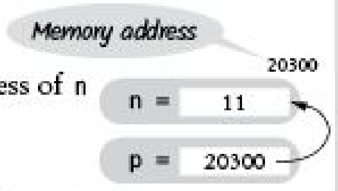
```
vector<string> names;
```

Add elements to the end

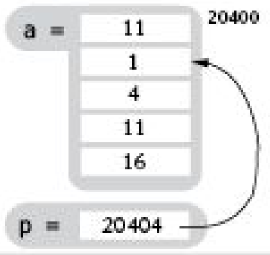
```
names.push_back("Ann");
names.push_back("Cindy"); // names.size() is now 2

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);
```

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

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)

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);     // Constructor declaration
   void deposit(double amount);    // Member function declaration
   double get_balance() const;     // Accessor member function
   ...
private:
   double balance;                 // Data member
};

void BankAccount::deposit(double amount)   // Member function definition
{
   balance = balance + amount;
}
```

Inheritance

Derived class *Base class*

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

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