

Name:										
EmpID:										

## 1 (30 points) Answer the following questions.

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

- (2) Define function header for function **sort**, for two strings `a` and `b`, if `a` has more characters than `b`, swap them. Return type is `void`.

- (3) Write a statement to generate a random **floating point number** in `[0, 2]`. 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(2, 6)`?

- (5) What is the value of `1 + 89 / 10`?

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

```
int foo(int n) {  
    int sum = 0;  
    for (int i = 1; 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  $\frac{\sqrt{b}}{2a}$ . You may use `sqrt` function, see cheat sheet.

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

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int count = 0;
6     for (int i = 10; i > 0; i -= 3)
7         count++;
8
9     cout << count << endl;
10    return 0;
11 }
```

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

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 void foo(int arr[], int size);
6 int main() {
7     int arr[] = {3, 1, 2};
8     int size = sizeof(arr) / sizeof(arr[0]);
9     foo(arr, size);
10
11     for (int i = 0; i < size; i++)
12         cout << arr[i] << " ";
13
14     return 0;
15 }
16
17 void foo(int arr[], int size) {
18     for (int j = size-1; j >= 1; j--)
19         for (int i = 0; i < j; i++)
20             if (arr[i] < arr[i+1])
21                 swap(arr[i], arr[i+1]);
22 }
```

(10) Write a condition to represent that integer  $i$  is larger than 0 **and**  $i$  is even. An integer is even if it can be divided by 2.

## 2 (20 points) Answer the following questions.

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

```
1  #include <iostream>
2  using namespace std;
3
4  void show(int size);
5  int main() {
6      show(4);
7      return 0;
8  }
9
10 void show(int size) {
11     for (int row = 0; row < size; row++) {
12         for (int col = 0; col < size; col++)
13             if (row + col == size-1)
14                 cout << "*";
15             else cout << "-";
16
17         cout << endl;
18     }
19 }
```

(2) Define function `no_digit`, for a string, return true if **none** of its elements is a digit character, that is, one from '0' to '9', otherwise, return false. For example, `no_digit("abc")` returns true and `no_digit("ab1")` returns false. **No need to include libraries or define main function.**

Hint: you may use `int isdigit(int ch)` to test whether a character is digit letter '0' - '9' 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 **larger than** the target to the size of the array.

For example, if the array has elements 3, 2, -1, and target is 1, then there are 2 integers out of 3 elements that is larger than the target, so the percentage is 66.6667%. The return is 66.6667, 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 3 and 5 at the same time, the number of integers that can be divided by 3 only, and the number of integers that can be divided by 5 only.

Integer num is divided by 3 means the remainder of num divided by 3 is 0.

```
Enter an int (-1 to stop): 5
Enter an int (-1 to stop): 3
Enter an int (-1 to stop): 15
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 3 and 5: 2
number can be divided by 3 only: 2
number can be divided by 5 only: 1
```



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

## Selected Operators and Their Precedence

(See Appendix B for the complete list.)

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

## Loop Statements

```
while (balance < TARGET)
{
    year++;
    balance = balance * (1 + rate / 100);
}
```

Executed while condition is true

```
for (int i = 0; i < 10; i++)
{
    cout << i << endl;
}
```

```
do
{
    cout << "Enter a positive integer: ";
    cin >> input;
}
while (input <= 0);
```

Loop body executed at least once

## Conditional Statement

```
if (floor >= 13)
{
    actual_floor = floor - 1;
}
else if (floor >= 0)
{
    actual_floor = floor;
}
else
{
    cout << "Floor negative" << endl;
}
```

Condition

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

```
double cube_volume(double side_length)
{
    double vol = side_length * side_length * side_length;
    return vol;
}
```

Return type

Parameter type and name

Exits function and returns result.

```
void deposit(double& balance, double amount)
{
    balance = balance + amount;
}
```

Reference parameter

Modifies supplied argument

## Arrays

```
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]
}
```

Element type

Length

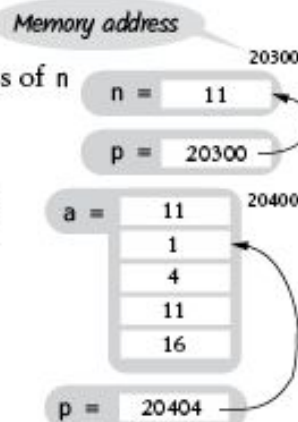
## Vectors

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

`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*  
`}`

`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*  
`}`

## Static Variables

### Static Data Members

## Range-based for Loop

`for (int v : values)` *An array, vector, or other container (C++ 11)*  
{  
    `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:` *Data member*  
        `double balance;`  
};  
`void BankAccount::deposit(double amount)` *Member function definition*  
{  
    `balance = balance + amount;`  
}

## Inheritance

`class CheckingAccount : public BankAccount` *Derived class*  
{  
    `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++;`  
}