**CS1410-40 2020 Fall Midterm 1**

1 **(5 points)**. What is the difference between switch statement and if-else statement? For example:

// version 1: switch statement, variable is an integer

switch(variable) {

case 1:

variable = 9;

break;

case 2:

variable = 17;

break;

case 3:

variable = 4;

break;

default:

variable = 0;

break;

}

// version 2: if-else statement, variable is an integer

if(variable == 1) {

variable = 9;

}

else if(variable == 2) {

variable = 17;

}

else if(variable == 3) {

variable = 4;

}

else {

variable = 0;

}

2 **(5 points)**. What is the keyword auto for?

3 **(5 points)**. Can a program be compiled without a main function?

4 **(5 points)**. What is the difference between postfix increment and prefix increment? For example:

int i = 0;

int a = i++; // postfix increment

int b = ++i; // prefix increment

5 **(5 points)**. What is the difference between “call by value” and “call by reference” when passing a parameter to a function?

void call\_by\_value(int a) {

// …

}

void call\_by\_reference(int& a) {

// …

}

6 **(5 points)**. What is the purpose of srand(time(0))? Why do we need it?

7 **(5 points)**. What is an ASCII table?

8 **(5 points)**. Please summarize the difference between: (1) compiled languages, (2) interpreted languages, and (3) just-in-time compilation.

9 **(5 points)**. What is the difference between: (1) std::cout << “hello world” and (2) std::cout << “hello” << ‘ ‘ << “world”?

10 **(5 points)**. What is wrong with the following loop for printing a list of numbers, 10, 9, 8, …, 0?

for(unsigned i=10; i>=0; i--) {

std::cout << i << ‘\n’;

}

Write down your fix using for-loop statement below:

11 **(5 points)**. Given a binary array (i.e., contains only 0 and 1), find the maximum number of consecutive 1s in this array. For example:

Input: [1,1,0,1,1,1]

Output: 3

The first two digits or the last three digits are consecutive 1s. The maximum number of consecutive 1s is 3.

int max\_consecutive\_ones(vector<int>& nums) {

// TODO

}

12 **(5 points)**. A rotation (or circular shift) is an operation similar to shift except that the bits that fall off at one end are put back to the other end. In left rotation, the bits that fall off at left end are put back at right end. In right rotation, the bits that fall off at right end are put back at left end. For example, let *n* is stored using 8 bits:

Left rotation of *n* = 11100101 by 3 makes *n* = 00101111 (Left shifted by 3 and first 3 bits are put back in last).

Right rotation of n = 11100101 by 3 makes n = 10111100 (Right shifted by 3 and last 3 bits are put back in first.

int leftRotate(int n, unsigned int d) {

// TODO

}

int rightRotate(int n, unsigned int d) {

// TODO

}

13 **(10 points)**. Given a limited range array of size N where the array contains elements between 1 to n-1 with one duplicate number, write a program to find the duplicate number. For example

Input: {1, 2, 4, 3, 4}

Output: 4

Input: {1, 2, 1, 3, 4, 6, 5}

Output: 1

int duplicate\_number(std::vector<int>& array) {

}

14 **(15 points)**. In lecture 12, we wrote a program of using “two-level for-loop” to find the maximum subarray sum (i.e., contiguous array sum) over an array of integer numbers. For example:



The solution of using two-level for-loop is essentially “brute-force” since we enumerate all possible length of the subarray. The number of required iterations is approximately *N2* where *N* is the size of the array, and it is very slow when the array is large. Your job is to solve this problem again ***using only one level of for-loop*** such that the number of iterations you need is no larger than *N*.

int max\_subarray\_sum(const std::vector<int>& array) {

// TODO

}

int main() {

std::vector<int> array = {-2, -3, 4, -1, -2, 1, 5, -3};

std::cout << max\_subarray\_sum(array) << ‘\n’;

}

15 **(15 points)**. Merge Sort is one of the most powerful sorting algorithms using divide and conquer. It iteratively partitions the array by half and combines two subproblems into a single sorted array. For example:



Your job is to complete the following merge\_sort function that takes a reference to an input array of integer numbers and sort them. You are free to declare other functions you need.

void merge\_sort(std::vector<int>& array) {

// TODO

}

int main() {

std::vector<int> array = {20, 90, 40, 30, 80, 70, 50};

merge\_sort(array);

for(auto i : array) { // sorted numbers

std::cout << i << ‘\n’;

}

}