COSC 1550 Programming 1

# Practice Exam

Please record the time you used for these questions.
The time limit for the final exam is about 80~100 minutes.
So please try to complete this practice exam in 90 minutes.
Please complete this practice exam before next Thursday.

Part I
Predict the output of the following code segments. (20%)

(1)

```
int x = 10,
    y = 15;
double a = 2.0,
       b;
b = x / y * a;
cout << b << endl;
```

*0*

(2)

```
int x = 10,
    y = 15;
double a = 2.0,
       b;
b = a * x / y;
cout << b << endl;
```

*1.33333*

(3)

```
int a = 10,
    b = 20;

if (a == 10)
{
    cout << 'A';
}
else if (b == 20)
{
    cout << 'B';
}
else
{
    cout << '?';
}
```

*A*

(4)

```
int a = 5 / 2;

switch (a)
{
    case 0:
        cout << 'a';
        break;
    case 1:
        cout << 'b';
        break;
    default:
        cout << 'c';
}
```

*c*

(5)

```
const int WIDTH = 3,
          HEIGHT = 4;
int matrix[HEIGHT][WIDTH] = { { 1, 2, 3},
                              { 4, 5, 6},
                              { 7, 8, 9},
                              {10,11,12}};

cout << matrix[2][1] << endl;
cout << matrix[3][2] << endl;
```

*7*

*12*

(6)

```
for (int x = 0; x < 3; x++)
{
    for (int y = 7; y > 5; y--)
    {
        cout << x << y << endl;
    }
}
```

*07*
*06*
*17*
*16*
*27*
*26*

(7)

```
int a = 1;

while (a < 10)
{
    if (a % 3 == 0)
    {
        a *= 3;
    }
    else
    {
```

*2*
*3*
*9*
*27*

```
        a++;
    }
    cout << a << endl;
}
```

(8)

```
int sum = 0,
    numbers[8] = { 3, 6, 8, 2, 5, 8, 7, 1 };

for (int x = 2; x < 5; x++)
{
    sum += numbers[x];
}
cout << sum;
```

*15*

(9)

```
int x = -1,
    arr[7] = { 9, 7, 6, 3, 8, 2, 5 };

for (int i = 0; i < 7; i++)
{
    if (arr[i] % 2 == 0)
    {
        x = i;
    }
}
cout << x;
```

*5*

(10)

```
int x = 1,
    y = 2;

do
{
    int x = 0;
    x = x + y;
    y--;
} while (y > 0);

cout << x;
```

*1*

Part II
Searching and sorting
(1)
Given these numbers in an array: 112, 135, 256, 257, 355, 364, 397, 401
If we use binary search to search for the target number 255,
show the steps of the comparisons. (5%)
(E.g. the first step is we compare 257 with 255)

*257 larger than 255*

*135 less than 255*

*256 larger than 255*

*Not found.*

(2)
Given these numbers in an array: 2, 4, 1, 3, 5
If we use bubble sort to sort this array in ascending order,
show the steps of how the array changes. (5%)

*2 1 4 3 5*

*2 1 3 4 5*

*1 2 3 4 5*

(3)
Complete the function of linear search (5%)

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

int search(const char arr[], int size, char target);      // fill the blank

int main()
{
    const int SIZE = 9;
    int targetindex;
    char target = 'G',
         letterset[SIZE] = {'B', 'M', 'P', 'J', 'P', 'G', 'T', 'I', 'F'};

    targetindex = search(letterset, SIZE, target);       // fill the blank

    if (targetindex == -1)
        cout << "The target is not found." << endl;
    else
        cout << "The target is found at location "
             << targetindex + 1 << endl;

    return 0;
}

int search(const char arr[], int size, char target)  // complete the function
{
    bool found = false;
    int targetindex = -1;

    for (int i = 0; (i < size) && (!found); i++)
    {
        if (arr[i] == target)
        {
            found = true;
            targetindex = i;
        }
    }

    return targetindex;
}
```

Part III
Write functions. (20%)
Write a function (function definition) for each of the questions below.
For example, write a function to compute and return the sum of 2 floating point numbers.

```
float computeSum(float x, float y)
{
    return x + y;
}
```

(1)
Write a function to compute the area of a rectangle. The function takes 2 double parameters: width and height. Then the function computes the area (width x height) and return it.

```
double compute_area(double width, double height)
{
    return width * height;
}
```

(2)
Write a function. The function asks the user to type a floating point number, and then "return" its square and cube.

```
void compute_square_and_cube(float number, float & n2, float & n3)
{
    n2 = number * number;
    n3 = number * n2;
}
```

(3)
Write a function. The function takes an integer array as its parameter, and it returns the number of odd numbers in the array.
For example, if the array is {5, 6, 7, 8, 9}, the function returns 3, because there are 3 odd numbers: 5, 7, and 9. (Hint: How can the function know the size of the array?)

```
int count_odd_numbers(const int arr[], int size)
{
    int num_of_odds = 0;

    for (int i = 0; i < size; i++)
    {
        if (arr[i] % 2 == 1)
            num_of_odds++;
    }

    return num_of_odds;
}
```

(4)
Write a function. The function takes 2 float arrays as it parameters. (The two arrays may have different sizes.) Then the function shall return the total sum of all elements of the arrays.
For example, if the arrays are {1, 2, 3} and {4, 5}, the function returns 15.

```
float compute_sum(const float arr1[], int size1,
                  const float arr2[], int size2)
{
    float sum = 0.0f;

    for (int i = 0; i < size1; i++)
    {
        sum += arr1[i];
    }

    for (int i = 0; i < size2; i++)
    {
        sum += arr2[i];
    }

    return sum;
}
```

Part IV
Tracing functions
Predict the output of the following programs. (20%)
(1)

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

int dosomething1(int a, int b, int c);
int dosomething2(int a, int b);

int main()
{
    int a = 5,
        b = 4,
        c = 3,
        d;

    d = dosomething1(3, 2, 1);
    cout << d << endl;

    d = dosomething1(d, a + 3, b);
    cout << d << endl;

    d = dosomething1(a + 1, b, c - 1);
    cout << d << endl;

    return 0;
}

int dosomething1(int a, int b, int c)
{
    return a + dosomething2(b, c);
}

int dosomething2(int a, int b)
{
    return a / b;
}
```

```
5
7
8
```

(2)

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

void change_reference(int &, int);

int main()
{
    int x = 5,
        y = 7;

    change_reference(x, y);
    cout << x << y << endl;
    change_reference(y, x); // read carefully!
    cout << x << y << endl;

    return 0;
}

void change_reference(int & x, int y)
{
    x++;
    y--;
    cout << x << y << endl;
}
```

```
66
67
85
68
```

(3)

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

void change_array(int arr[], int size);

int main()
{
    const int LENGTH = 5;
    int arr[LENGTH] = {5, 4, 3, 2, 1};

    change_array(arr, 3);

    for (int i = 0; i < LENGTH; i++)
    {
        cout << arr[i] << endl;
    }

    return 0;
}

void change_array(int arr[], int size)
{
    for (int i = 0; i < size; i++)
    {
        arr[i] *= 2;
    }
}
```

```
10
8
6
2
1
```

(4)

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

void copy_array(int arr1[], int arr2[], int size);

int main()
{
    int arr[5] = {5, 4, 3, 2, 1},
        num[7] = {3, 4, 5, 6, 7, 8, 9};

    copy_array(arr, num, 4);

    for (int i = 0; i < 7; i++)
    {
        cout << num[i] << endl;
    }

    return 0;
}

void copy_array(int arr1[], int arr2[], int size)
{
    for (int i = 0; i < size; i++)
    {
        arr2[i] = arr1[i];
    }
}
```

```
5
4
3
2
7
8
9
```

Part V
Writing a program.
The program has an array price that holds the unit price of each fruit. price[0] is the unit price of apple. price[1] is the unit price of banana. price[2] is the unit price of grapefruit. price[3] is the unit price of watermelon. The price of each fruit is fixed.

The program has an array quantity that holds the quantity of each fruit. quantity[0] is the quantity of apple. quantity[1] is the quantity of banana. quantity[2] is the quantity of grapefruit. quantity[3] is quantity of watermelon. The quantity of each fruit is to be entered by the user.

Write a program to compute the total price of these 4 fruits.

Requirements:
(1) Your program should have a function get_quantity(), which asks the user to type the quantities of the 4 fruits. (10%)
(2) Your program should have a function compute_price(), which computes the total price of the 4 fruits. The function should take the above 2 arrays as the parameter and return the total price. (10%)
(3) The main function should call the above 2 functions and then display the total price on the command prompt. (5%)
You do not need to consider the situation when the user types a negative number or a non-numerical value.

Part of the program has been provided:

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

void get_quantity(int qtty[], int size);
double compute_price(const double prc[], const int qtty[], int size);

int main()
{
    const int NUM_OF_FRUITS = 4;
    const double price[NUM_OF_FRUITS] = {1.49, 0.35, 0.99, 4.99};
    double totalprice;
    int quantity[NUM_OF_FRUITS];

    get_quantity(quantity, NUM_OF_FRUITS);
    totalprice = compute_price(price, quantity, NUM_OF_FRUITS);

    cout << "The total price is " << totalprice << endl;

    return 0;
}

void get_quantity(int qtty[], int size)
{
    cout << "Please enter the quantities of the four fruits:" << endl;
    for (int i = 0; i < size; i++)
    {
        cin >> qtty[i];
    }
}

double compute_price(const double prc[], const int qtty[], int size)
{
    double totalprice = 0.0;

    for (int i = 0; i < size; i++)
    {
        totalprice += prc[i] * qtty[i];
    }

    return totalprice;
}
```