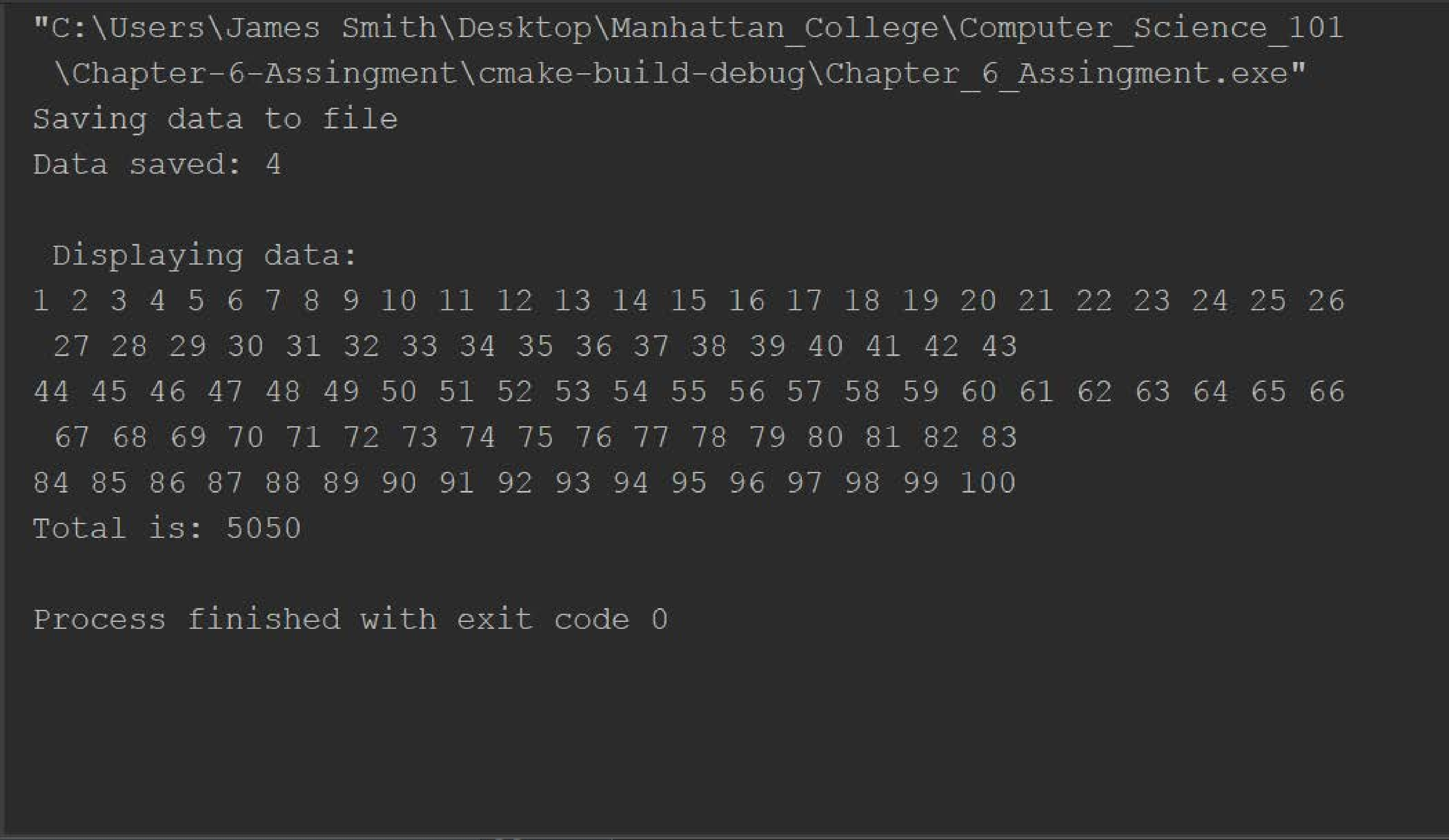
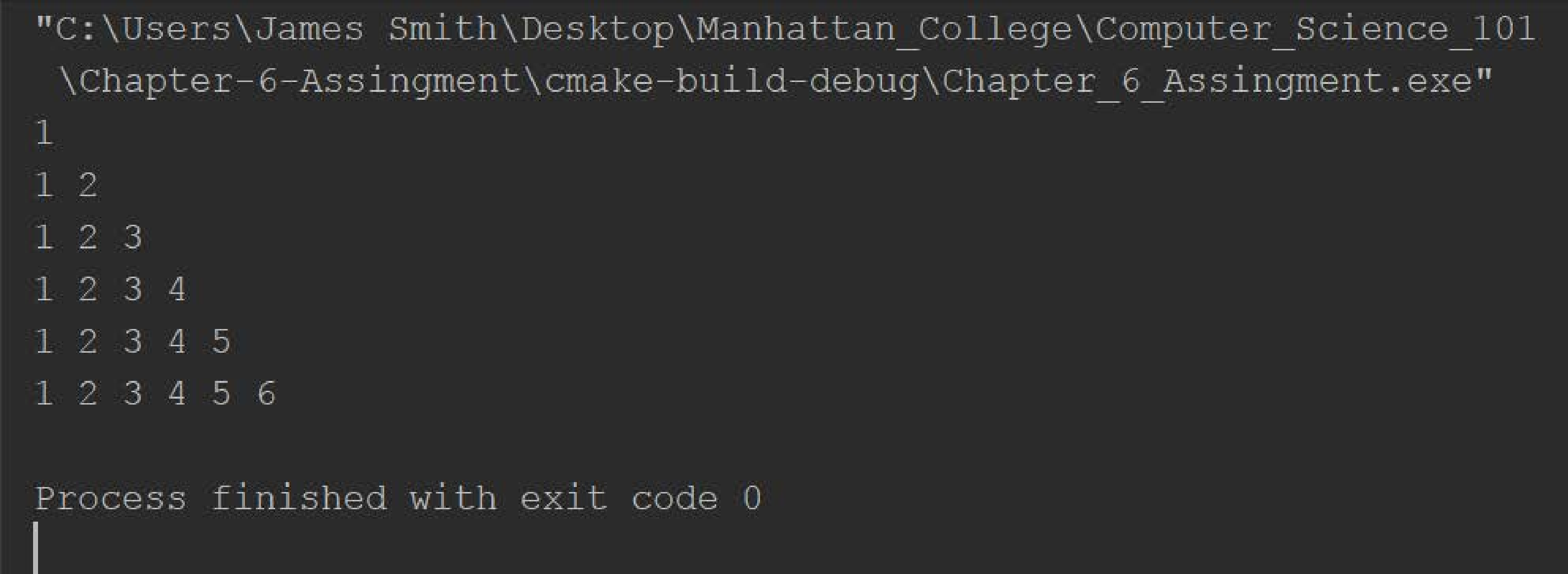
File - C:\Users\James Smith\Desktop\Manhattan_College\Computer_Science_101\Chapter-6-Assingment\Source-Files\main.cp

```
#include <iostream>
#include <fstream>

using namespace std;

void create_file();
void output_data();
void display_data();

const string FILE_PATH = "..\\Text-Files\\number-file.txt";

int main() {
    create_file();
    output_data();
    display_data();
    return 0;
}

void create_file() {
    ofstream new_file{FILE_PATH};
    if (!new_file.is_open()){
        cout << "File not opened" << endl;
        exit(EXIT_FAILURE);
    }
    new_file.close();
}

void output_data() {
    ofstream output_file {FILE_PATH};
    int count{1};
    cout << "Saving data to file" << endl;
    for (int x = 1; x <= 100; x++) {
        output_file << x << endl;
        if (x == 100) { break; }
        count++;
    }
    cout << "Data saved: " << sizeof(count) << endl;
    output_file.close();
}

void display_data() {
    ifstream input_file{FILE_PATH};
    int sum {0};
    if (!input_file.is_open()) {
        cout << "The file could not be opened" << endl;
        exit(EXIT_FAILURE);
    }
    cout << "\n Displaying data: " << endl;
    string value;
    while (input_file >> value) {
        cout << value << " ";
        sum += stoi(value);
    }
    cout << "\nTotal is: " << sum << endl;

    input_file.close();
}


```



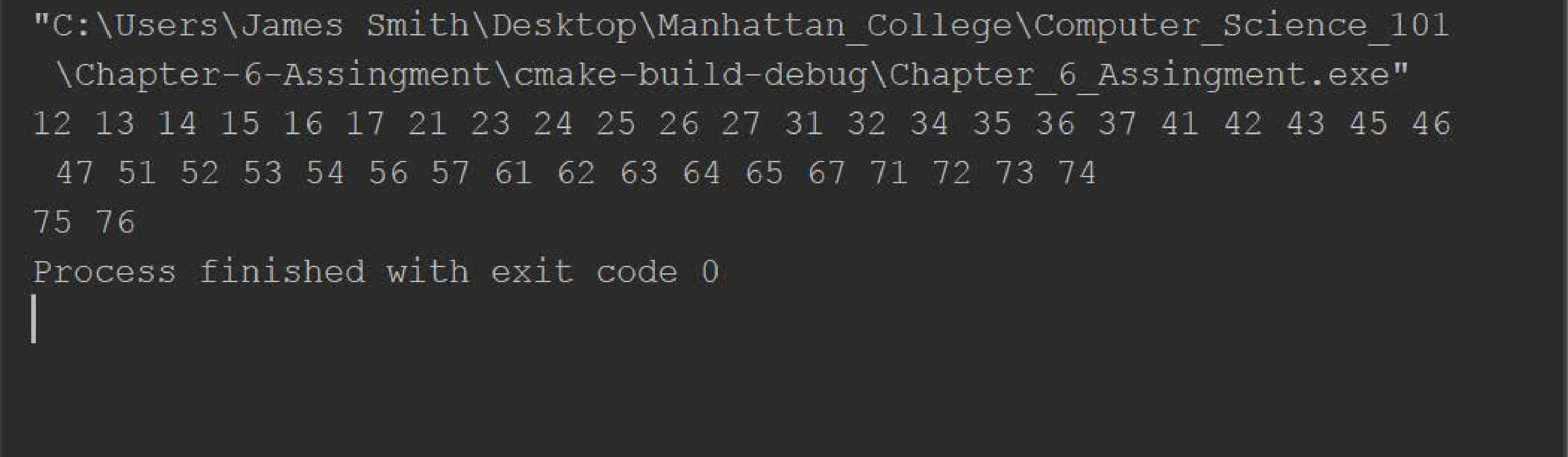
```
"C:\Users\James Smith\Desktop\Manhattan_College\Computer_Science_101
 \Chapter-6-Assingment\cmake-build-debug\Chapter_6_Assingment.exe"
Saving data to file
Data saved: 4

 Displaying data:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66
 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83
84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
Total is: 5050

Process finished with exit code 0
```

```cpp
#include <iostream>
#include <fstream>

using namespace std;

int main(){
    for (int row = 1; row <= 6; row++){
        for (int column = 1; column <= row; column++){
            cout << column << " ";
        }
        cout << endl;
    }
    return 0;
}
```



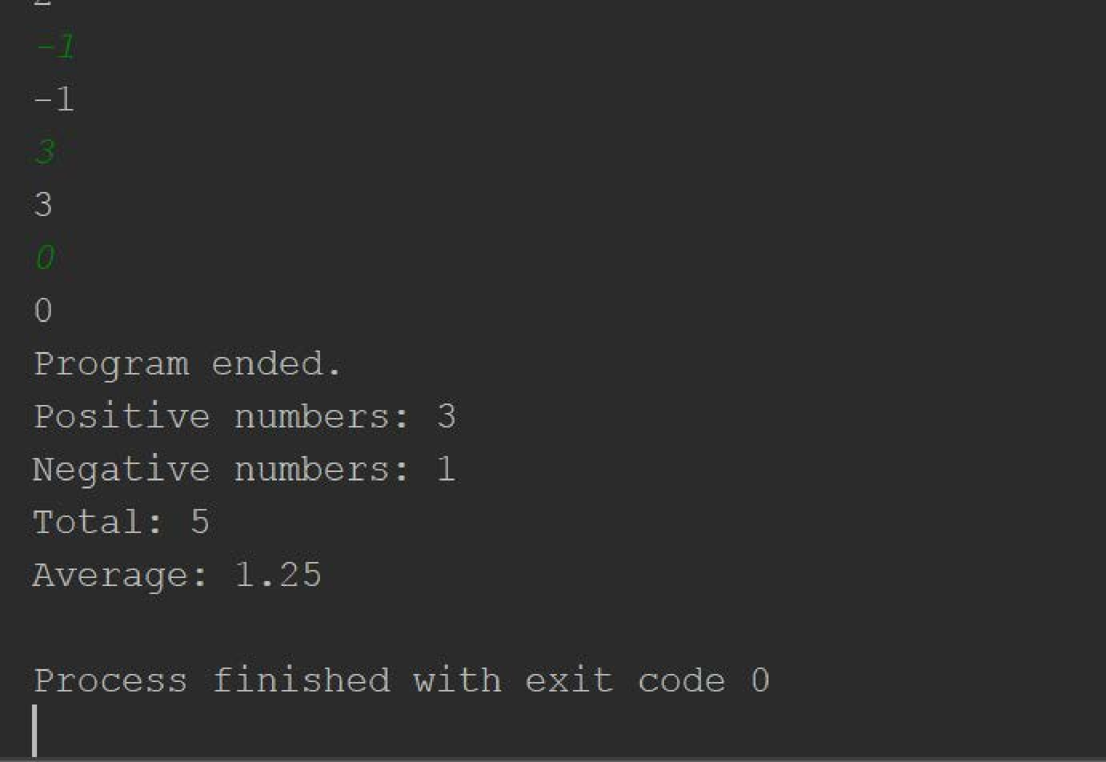
```
"C:\Users\James Smith\Desktop\Manhattan_College\Computer_Science_101
 \Chapter-6-Assingment\cmake-build-debug\Chapter_6_Assingment.exe"
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6

Process finished with exit code 0
```

```
#include <iostream>
#include <fstream>

using namespace std;

int main(){
    for (int x = 1; x <= 7; x++){
        for (int y = 1; y <= 7; y++){
            if (y == x){
                continue;
            }
            cout << x << y << " ";
        }
    }
    return 0;
}
```



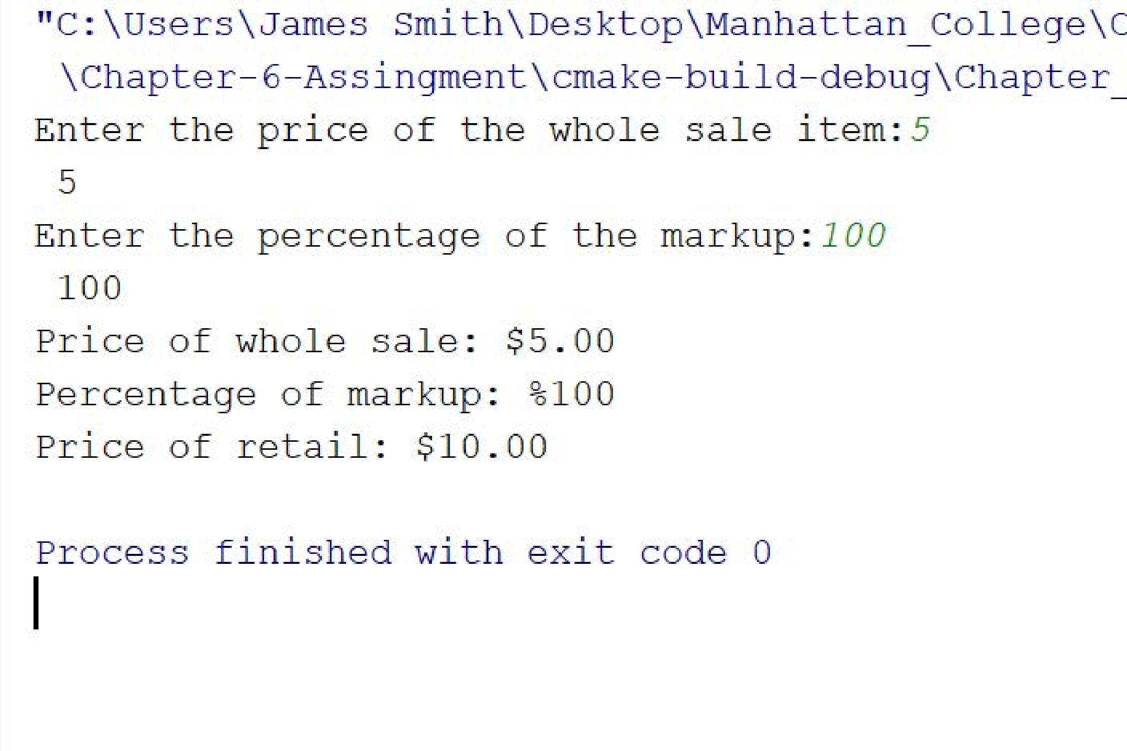
```
"C:\Users\James Smith\Desktop\Manhattan_College\Computer_Science_101
 \Chapter-6-Assingment\cmake-build-debug\Chapter_6_Assingment.exe"
12 13 14 15 16 17 21 23 24 25 26 27 31 32 34 35 36 37 41 42 43 45 46
 47 51 52 53 54 56 57 61 62 63 64 65 67 71 72 73 74
75 76
Process finished with exit code 0
```

```cpp
#include <iostream>

using namespace std;

int main() {
    float num, positive_num{0}, negative_num{0}, sum{0}, average{0};
    cout << "Note: Enter 0 to end the program." << endl;
    cout << "Enter the numbers you want to add: ";
    while (cin >> num) {
        if (num == 0) {
            cout << "Program ended. " << endl;
            break;
        }
        if (num > 0) positive_num++;
        if (num < 0) negative_num++;
        sum += num;
    }

    cout << "Positive numbers: " << positive_num << endl;
    cout << "Negative numbers: " << negative_num << endl;
    cout << "Total: " << sum << endl;
    average = sum / (positive_num + negative_num);
    cout << "Average: " << average << endl;
    return 0;
}
```



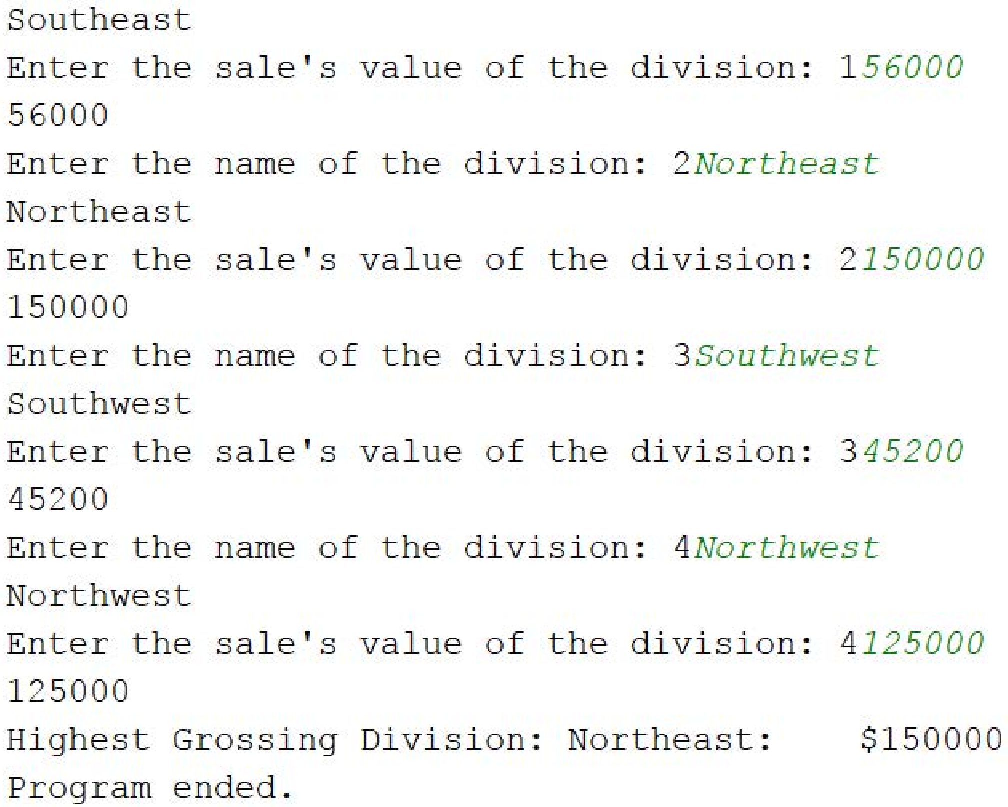
```
2
-1
-1
3
3
0
0
Program ended.
Positive numbers: 3
Negative numbers: 1
Total: 5
Average: 1.25

Process finished with exit code 0
```

```cpp
#include <iostream>
#include <typeinfo>
#include <cmath>

using namespace std;

double cal_retail(double whole_sale_price, double stock_percentage){
    double decimals{0};
    double retail_price{0};

    if (whole_sale_price < 0 || stock_percentage < 0){
        cout << "Sale price or markup cannot be less than $0 and %0" << endl;
        exit(EXIT_FAILURE);
    }else if (whole_sale_price == 0 || stock_percentage == 0){
        cout << "Values could not be calculated. " << endl;
        exit(EXIT_SUCCESS);
    }

    if (modf(stock_percentage, &decimals) == 0.0){
        stock_percentage /= 100;
    }

    retail_price = whole_sale_price + (whole_sale_price * stock_percentage);

    return  retail_price;
}

int main() {
    double whole_pri, markup_pr;
    cout << "Enter the price of the whole sale item: ";
    cin >> whole_pri;
    cout << "Enter the percentage of the markup: ";
    cin >> markup_pr;
    double retail_price {cal_retail(whole_pri, markup_pr)};
    printf("Price of whole sale: $%.2f \n", whole_pri);
    printf("Percentage of markup: %%%.0f \n", markup_pr);
    printf("Price of retail: $%.2f \n", retail_price);

    return 0;
}
```



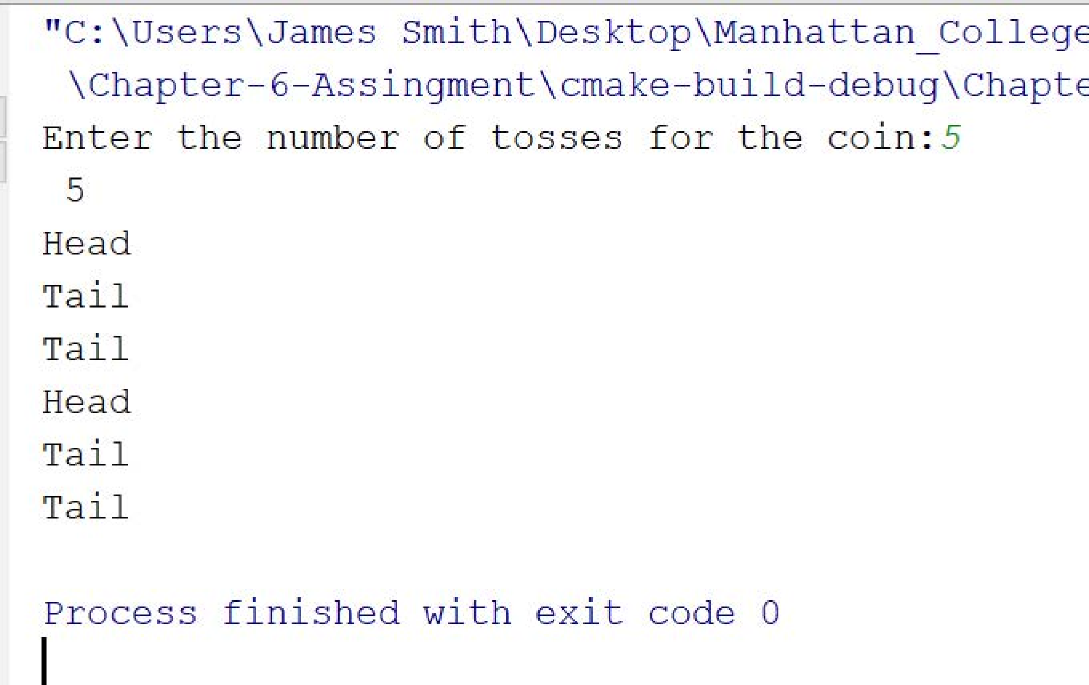
```
"C:\Users\James Smith\Desktop\Manhattan_College\C
 \Chapter-6-Assingment\cmake-build-debug\Chapter_
Enter the price of the whole sale item:5
 5
Enter the percentage of the markup:100
 100
Price of whole sale: $5.00
Percentage of markup: %100
Price of retail: $10.00


Process finished with exit code 0
```

```cpp
#include <iostream>
#include <array>

const unsigned int ARR_SIZE = 10;

void get_sales(std::string division_repo[], double quarter_profits_repo[], unsigned &division_count) {

    std::cout << "Note: The size of the Array is 10" << std::endl;
    std::cout << "How many divisions will be saved?: ";
    std::cin >> division_count;

    if (division_count <= 0 || division_count > ARR_SIZE) {
        if (division_count == 0) {
            std::cout << "You have entered a value of 0" << std::endl;
            exit(EXIT_SUCCESS);
        } else {
            std::cout << "The value entered cannot be evaluated." << std::endl;
            exit(EXIT_FAILURE);
        }
    } else {
        unsigned counter{1};
        unsigned tries{1};
        std::string div_name;
        double div_sales{0};

        while (counter <= division_count) {

            std::cin.ignore();
            std::cout << "Enter the name of the division: " << counter;
            std::getline(std::cin, div_name);
            enter_sales:
            std::cout << "Enter the sale's value of the division: " << counter;
            std::cin >> div_sales;

            if (div_sales < 0) {
                std::cout << "Invalid input" << std::endl;
                std::cout << "Press any key to continue: ";
                std::cout << "Tries: " << tries << std::endl;

                if (tries >= 3) {
                    std::cout << "The program has ended." << std::endl;
                    exit(EXIT_SUCCESS);
                }

                std::cin.ignore();
                std::cin.get();
                std::cout << "Try again: ";
                tries++;
                goto enter_sales;
            }

            division_repo[counter - 1] = div_name;
            quarter_profits_repo[counter - 1] = div_sales;
            counter++;
        }
    }
}

void get_highest(std::string divs[], double div_sales[], unsigned list_size) {
    if (divs != nullptr && div_sales != nullptr) {
        double current{0};
        double next{0};

        std::string current_name;
        std::string next_name;


        for (int index = 0; index < list_size; index++) {
            for (int index_t = 0; index_t < list_size - 1; index_t++) {
                if (div_sales[index_t] > div_sales[index_t + 1]) {
                    current = div_sales[index_t];
                    next = div_sales[index_t + 1];
                    div_sales[index_t + 1] = current;
                    div_sales[index_t] = next;
                    current_name = divs[index_t];
                    next_name = divs[index_t + 1];
                    divs[index_t + 1] = current_name;
                    divs[index_t] = next_name;
                } else {
                    continue;
                }
            }

            if (index == list_size - 1) {
                printf("Highest Grossing Division: %s: \t $%.f \n", divs[index].c_str(), div_sales[index]);
            }
        }

        std::cout << "Program ended." << std::endl;

    } else {
        std::cout << "The repository is empty. " << std::endl;
        exit(EXIT_FAILURE);
    }
}

int main() {
    std::string divs[ARR_SIZE];
    double sales[ARR_SIZE];
    unsigned list_size;
    get_sales(divs, sales, list_size);
    get_highest(divs, sales, list_size);

    return 0;
}
```



```
Southeast
Enter the sale's value of the division: 156000
56000
Enter the name of the division: 2Northeast
Northeast
Enter the sale's value of the division: 2150000
150000
Enter the name of the division: 3Southwest
Southwest
Enter the sale's value of the division: 345200
45200
Enter the name of the division: 4Northwest
Northwest
Enter the sale's value of the division: 4125000
125000
Highest Grossing Division: Northeast:    $150000
Program ended.
```

```cpp
#include <iostream>
#include <map>
#include <string>
#include <functional>
#include <ctime>

using namespace std;

//toss_coin function
void toss_coin(unsigned);

int main () {
    unsigned num_of_toss;
    cout << "Enter the number of tosses for the coin: ";
    cin >> num_of_toss;
    toss_coin(num_of_toss);
    return 0;
}

//Toss coin
void toss_coin(unsigned toss_count){
    if(toss_count <= 0){
        cout << "Coin simulation ended." << endl;
        exit(EXIT_FAILURE);
    }
    unsigned toss_counter {0};
    unsigned seed{static_cast<unsigned int>(time(nullptr))};
    srand(seed);
    while (toss_counter <= toss_count){
        unsigned toss{static_cast<unsigned int>(rand() % 2 + 1)};
        if (toss == 1){
            cout << "Head" << endl;
        }else if (toss == 2){
            cout << "Tail" << endl;
        }
        toss_counter++;
    }
}

```



```
"C:\Users\James Smith\Desktop\Manhattan_College
 \Chapter-6-Assingment\cmake-build-debug\Chapte
Enter the number of tosses for the coin:5
 5
Head
Tail
Tail
Head
Tail
Tail

Process finished with exit code 0
```