

```

1 #include <iostream>
2 #include <array>
3
4 const unsigned int ARR_SIZE = 10;
5
6 void get_sales(std::string division_repo[], double quarter_profits_repo[],
    unsigned &division_count) {
7
8     std::cout << "Note: The size of the Array is 10" << std::endl;
9     std::cout << "How many divisions will be saved?: ";
10    std::cin >> division_count;
11
12    if (division_count <= 0 || division_count > ARR_SIZE) {
13        if (division_count == 0) {
14            std::cout << "You have entered a value of 0" << std::endl;
15            exit(EXIT_SUCCESS);
16        } else {
17            std::cout << "The value entered cannot be evaluated." << std::endl;
18            exit(EXIT_FAILURE);
19        }
20    } else {
21        unsigned counter{1};
22        unsigned tries{1};
23        std::string div_name;
24        double div_sales{0};
25
26        while (counter <= division_count) {
27
28            std::cin.ignore();
29            std::cout << "Enter the name of the division: " << counter;
30            std::getline(std::cin, div_name);
31            enter_sales:
32            std::cout << "Enter the sale's value of the division: " << counter;
33            std::cin >> div_sales;
34
35            if (div_sales < 0) {
36                std::cout << "Invalid input" << std::endl;
37                std::cout << "Press any key to continue: ";
38                std::cout << "Tries: " << tries << std::endl;
39
40                if (tries >= 3) {
41                    std::cout << "The program has ended." << std::endl;
42                    exit(EXIT_SUCCESS);
43                }
44
45                std::cin.ignore();
46                std::cin.get();
47                std::cout << "Try again: ";
48                tries++;
49                goto enter_sales;
50            }
51
52            division_repo[counter - 1] = div_name;
53            quarter_profits_repo[counter - 1] = div_sales;
54            counter++;
55        }
56    }
57 }
58
59 void get_highest(std::string divs[], double div_sales[], unsigned list_size) {

```

```

60     if (divs != nullptr && div_sales != nullptr) {
61         double current{0};
62         double next{0};
63
64         std::string current_name;
65         std::string next_name;
66
67
68         for (int index = 0; index < list_size; index++) {
69             for (int index_t = 0; index_t < list_size - 1; index_t++) {
70                 if (div_sales[index_t] > div_sales[index_t + 1]) {
71                     current = div_sales[index_t];
72                     next = div_sales[index_t + 1];
73                     div_sales[index_t + 1] = current;
74                     div_sales[index_t] = next;
75                     current_name = divs[index_t];
76                     next_name = divs[index_t + 1];
77                     divs[index_t + 1] = current_name;
78                     divs[index_t] = next_name;
79                 } else {
80                     continue;
81                 }
82             }
83
84             if (index == list_size - 1) {
85                 printf("Highest Grossing Division: %s: \t %.f \n", divs[index
86 ].c_str(), div_sales[index]);
87             }
88
89             std::cout << "Program ended." << std::endl;
90
91         } else {
92             std::cout << "The repository is empty. " << std::endl;
93             exit(EXIT_FAILURE);
94         }
95     }
96
97 int main() {
98     std::string divs[ARR_SIZE];
99     double sales[ARR_SIZE];
100     unsigned list_size;
101     get_sales(divs, sales, list_size);
102     get_highest(divs, sales, list_size);
103
104     return 0;
105 }

```