```cpp
//
//  main.cpp
//  A2_COSC_1437_014
//
//  Created by Nicholas Brown on 1/21/25.
//


#include <iostream>
using namespace std;

double median(int *arr, int size);
int mode(int *arr, int size);
int *makeArray(int arr); // Did not use but will ask the teacher for guidance.
void getMovieData(int *arr, int size);
void selectionSort(int arr[], int size);
double average(int *arr, int size);

// Added personal functions below.
void swap (int &a, int &b);
bool isEven (int num);

int main() {

    int size;
    cout << "Number of students surveyed: ";
    cin >> size;

//declare the pointer into the dynamic array.
    int *arr = new int[size];
// new - allocates memory of whatever it is pointing to and returns the memory which is originally stored by the variable array.

    getMovieData(arr, size);



    selectionSort(arr, size);

    cout << "Average: " << average(arr,size) << endl;

    cout << "Median: " << median(arr,  size) << endl;

    cout << "Mode: " << mode(arr, size) << endl;



// Deallocates memory for program efficientness.
    delete [] arr;
    arr = nullptr;
```

```
23  int main() {
53      return 0;
54  }
55
56
57
58  // Function to get user's data.
59  void getMovieData(int *arr, int size){
60
61
62
63      cout << "Number of movies seen by each student: ";
64
65      for (int i = 0; i < size; i++) {
66              cin >> *(arr + i);
67          }
68
69
70
71
72  }
73
74
75  // Function to get user's average.
76  double average(int *arr, int size){
77
78
79      int sum = 0;
80      double avg = 0.0;
81
82      for (int i = 0; i < size; i++) {
83          sum += *(arr + i);
84      }
85
86
87      //static cast is needed for me to return a double value because if i don't my function will return a integer value instead.
88      avg = static_cast<double>(sum) / size;
89
90
91      return avg;
92  }
93
94  // Function I personally added in to make the swapping functionality easier to read.
95  void swap(int& a, int& b){
96
97      int temp = a;
98      a = b;
99      b = temp;
100 }
101
102
```

```
103 // Function that is for the selection sort algorithm.
104 void selectionSort(int arr[], int size){
105
106     for (int i = 0; i < size; i++) {
107         int min_index = i;
108         for (int j = i + 1; j < size; ++j) {
109             if (arr[j] < arr[min_index]){
110                 min_index = j;
111             }
112
113         }
114         if (min_index != i) {
115             swap(arr[i], arr[min_index]);
116         }
117
118
119     }
120
121     // Testing to see if this sorts my array (delete later)
122     cout << "This is your array sorted: ";
123     for (int i = 0; i < size; i++) {
124         cout << arr[i] << " ";
125     }
126
127     cout << endl;
128 }
129
130 // Function to help differentiate between and even or odd dataset.
131 bool isEven (int num){
132
133     return num % 2 == 0;
134
135
136 }
137
138
139
140  //Function to get the user's median. odd size numbers work fine but even numbers need to be worked on further. W.I.P (01/26/25)
141 double median(int *arr, int size){
142     int result = size;
143
144     // Comparing my isEven function to differentiate if array size is even or odd.
145         if (isEven(result)) {
146             return (*(arr + size / 2 - 1) + *(arr + size / 2)) / 2.0;
147         } else {
148             return *(arr + size / 2);
149         }
150
151     }
152
153 int mode(int *arr, int size){
```

```
141 double median(int *arr, int size){
150
151     }
152
153 int mode(int *arr, int size){
154     int maxVal= 0;
155     int counter = 0;
156     int modeValue = *(arr + 0);
157
158     // the most reoccuring element in a set
159
160     for (int i = 0; i < size; i++) {
161         counter = 0;
162
163
164         for (int j = 0; j < size; j++){
165
166             if (arr[j] == arr[i]) {
167                 counter++;
168             }
169             // this defaults and looks at the next available element.
170             }
171
172
173         // So if the array element has a counter greater than the current maxVal I want to set it as the new maxVal and for where ever the array is stopped at I want to set it as the modeValue
                so that i can return the actual element instead of it's reoccurance.
174         if (counter > maxVal) {
175             maxVal = counter;
176             modeValue = arr[i];
177         }
178
179     }
180     // If the array has no mode I will output -1 for the user to see
181     if (maxVal == 1) {
182         return -1;
183     }
184
185     return modeValue;
186
187
188 }
189
190
191
192
193
194
```