Question 1

The two program *vector_sort.cpp* has 3 methods that are used for sorting and searching the vector. **vectorSort** returns a type vector¡int¿ and takes a vector¡int¿ as a parameter. It sorts the vector and returns the sorted vector in descending order. To accomplish this, it uses the method **swap** which takes the location of the two integer values within the vector to be swapped and the vector passed in as a reference. The boolean function **containsVal** takes a vector¡int¿ and the target integer as a parameter and returns true if the target is somewhere in the vector, and false otherwise. To find this value, the method performs a recursive binary search.

Sample I/O

```
Enter a value to search for:
```

10

```
10 was not found in the vector.
```

```
Enter a value to search for:
```

31

```
31 is in the vector!
```

Question 2

The first method called by *golf_scores.cpp* is **getScores**. This method takes no parameters, but is responsible for collection ten input values from the user and using them to fill a vector. This method will call **validInput** which takes the user input as a parameter and returns true if the input is acceptable and can be added to the vector, or false otherwise. Once the vector is assembled, the program calls **display** which takes the vector containing the scores and the average of all of the scores. The average is generated using **averageScore** which takes the scores as a vector and returns a double containing the average score.

Sample I/O

```
Enter up to 10 golf scores, type '-1' to terminate.
Score 1:
```

2

```
Score 2:
```

4

```
Score 3:
```

6

```
Score 4:
```

-1

```
Displaying all 3 golf scores:
2 4 6
The average score was 4

Enter up to 10 golf scores, type '-1' to terminate.
Score 1:
```

3

```
Score 2:
```

4

```
Score 3:
```

no

```
Invalid input, please try again.
Score 3:
```

-1

```
Displaying all 2 golf scores:
3 4
The average score was 3.5
```

Question 3

The first method called by *minimum_power.cpp* is **getPower**. This method takes no parameters and returns a 2-dimensional vector containing the values in the specified filename. Once this vector is established, the program calls **displayMin** which takes the vector as a parameter and displays both the minimum power for each week, and the day on which that value was seen.

Sample I/O

```
Getting data from power1.dat...

The lowest power in week 1 was 110.501 on day 1
The lowest power in week 2 was 169.643 on day 4
The lowest power in week 3 was 105.994 on day 2
The lowest power in week 4 was 101.868 on day 3
The lowest power in week 5 was 122.816 on day 7
The lowest power in week 6 was 220.780 on day 5
The lowest power in week 7 was 156.935 on day 4
The lowest power in week 8 was 183.840 on day 1
The lowest power in week 9 was 109.497 on day 4
The lowest power in week 10 was 103.516 on day 2
```

Question 4

The program *average_power.cpp* calls a similar method to **getPower** from the previous question. **getValues** performs a similar operation: it takes the values from a specified filename and returns a 2-dimensional vector containing those values. Once the values are stored appropriately in the vector, they are passed to **avgVec**, which returns a double value containing the average of all elements in the vector.

Sample I/O

```
Getting data from power1.dat...

The average value of the 2D vector read from power1.dat is: 292.7
```