w3e23.h

```
 1: // Programming in C/C++
 2: // Week 3: Assignment 23
 3: // Tjalling Otter &  Emiel Krol
 4: // Header file
 5:
 6: #ifndef W3E23FUNCTIONS
 7: #define W3E23FUNCTIONS
 8:
 9: #include <iostream>
10: #include <string>
11:
12: size_t partition(std::string* inputString, size_t left, size_t right);
13: bool lcIsSmaller(std::string inputStringA, std::string inputStringB);
14:
15: #endif
```

main.ih

```
 1: // Programming in C/C++
 2: // Week 3: Assignment 23
 3: // Tjalling Otter & Emiel Krol
 4: // Main file: internal header
 5:
 6: #include <iostream>
 7: #include <string>
 8:
 9: using namespace std;
10:
11: void quicksort(std::string* inputString, size_t left, size_t right);
12: size_t getEnvLength();
13: std::string *storeEnvStrings(size_t envLength);
14: bool lcIsSmaller(std::string inputStringA, std::string inputStringB);
```

main.cc

```
 1: // Programming in C/C++
 2: // Week 3: Assignment 23
 3: // Tjalling Otter & Emiel Krol
 4: // Main file
 5:
 6: #include "main.ih"
 7:
 8: int main()
 9: {
10:   const size_t envLength = getEnvLength();  // Get number of env. vars
11:
12:   string *envVars = storeEnvStrings(envLength); // Store them in a string array
13:
14:   quicksort(envVars, 0, envLength - 1); // Sort them alphabetically
15:
16:   for (size_t index = 0; index != envLength; ++index) // Print them (sorted)
17:   {
18:     cout << envVars[index] << "\n";
19:   }
20: }
```

getEnvLength.cc

```
 1: // Programming in C/C++
 2: // Week 3: Assignment 23
 3: // Tjalling Otter & Emiel Krol
 4: // Main file
 5:
 6: #include "w3e23.h"
 7:
 8: size_t getEnvLength()
 9: {
10:   extern char **environ;  // This gets the external array of environmental vars
11:   size_t index = 0;       // Initialise an integer
12:
13:   while (environ[index] != nullptr)     // For every non-null env. var (which the
14:     ++index;                          // last one is by definition), increment
15:                                       // index
16:   return index;                       // And return it
17: };
```

lcIsSmaller.cc

```
 1: // Programming in C/C++
 2: // Week 3: Assignment 23
 3: // Tjalling Otter & Emiel Krol
 4: // Lowercase compare (is A smaller than B) function
 5:
 6: #include "w3e23.h"
 7:
 8: bool lcIsSmaller(std::string inputStringA, std::string inputStringB)
 9: {
10:   for(auto& character : inputStringA)
11:     character = tolower(character);
12:
13:   for(auto& character : inputStringB)
14:     character = tolower(character);
15:   // For both input strings, convert them to lower case
16:
17:   if (inputStringA < inputStringB)
18:     return 1;
19:   // Then compare them as one would otherwise
20:
21:   return 0;
22: };
```

partition.cc

```
 1: // Programming in C/C++
 2: // Week 3: Assignment 23
 3: // Tjalling Otter & Emiel Krol
 4: // String array partition function
 5:
 6: #include "w3e23.h"
 7:
 8: size_t partition(std::string* inputString, size_t left, size_t right)
 9: {
10:   std::string pivot = inputString[left]; // Input left is starting string
11:   size_t pivotPosition = left;    // as well as its associated position
12:
13:                           // Starting from start string until end string
14:   for (size_t position = left; position <= right; ++position)
15:   {                       // If current string at position is smaller than pivot
16:     if (lcIsSmaller(inputString[position], inputString[pivotPosition]))
17:     {                     // Swap current string with one ahead of pivot string
18:                           // And swap pivot string and the one ahead it around
19:       swap(inputString[pivotPosition + 1], inputString[position]);
20:       swap(inputString[pivotPosition], inputString[pivotPosition + 1]);
21:       ++pivotPosition;  // Move on to next string as pivot element
22:     }
23:   }
24:   return (pivotPosition); // Return the pivot position
25: };
```

quicksort.cc

```
 1: // Programming in C/C++
 2: // Week 3: Assignment 23
 3: // Tjalling Otter & Emiel Krol
 4: // Quicksort function
 5:
 6: #include "w3e23.h"
 7:
 8: void quicksort(std::string* inputString, size_t left, size_t right)
 9: {
10:   if (left >= right)  // Quick return if left >= right
11:     return;
12:
13:   size_t mid = partition(inputString, left, right); // Partition and get pivot
14:   quicksort(inputString, left, mid);  // Sort string before pivot string
15:   quicksort(inputString, mid + 1, right); // Sort string after pivot string
16: };
```

storeEnvStrings.cc

```
 1: // Programming in C/C++
 2: // Week 3: Assignment 23
 3: // Tjalling Otter & Emiel Krol
 4: // Store environmental vars in string function
 5:
 6: #include "w3e23.h"
 7:
 8: std::string *storeEnvStrings(size_t envLength)
 9: {
10:   extern char **environ;
11:   std::string* envVars = new std::string[envLength];
12:   for (size_t index = 0; index != envLength; ++index)
13:     envVars[index] = environ[index];
14:   return envVars;
15: };
```