include.h

```c
#ifndef INCLUDE_H
#define INCLUDE_H

void Include(void) {
  char examples;  // used for user input

  printf("*** Include ***\n\n"
         "Include is a C preprocessor directive. It serves to incorporate\n"
         "the contents of another file into the current one. Usually, it\n"
         "is used to include header files from the C standard library or\n"
         "user defined ones. Include is usually placed at the start of a\n"
         "C file, before any of the other code. Included files can either\n"
         "be in the same directory as the file that includes them or can\n"
         "be added to the preprocessor using different flags depending\n"
         "on the compiler. Common header files are \n"
         "     stdio.h     // file and input/output functions and types\n"
         "     stdlib.h    // memory allocation, random\n"
         "     string.h    // see string function tutor\n"
         "     stdbool.h   // boolean type\n"
         "     math.h   // advanced math functions\n"
         "Header files are text files, typically ending with the .h extension.\n"
         "While they can contain full C code and definitions, they usually only\n"
         "contain declarations. The point is to provide different source files\n"
         "with public declarations for functions, variables, types, and more.\n"
         "The actual definitions are usually in a corresponding c file. The C\n"
         "code is then shared and used across all files that include the header,\n"
         "without recompiling the definitions each time.\n\n"
         "The syntax of include is \n"
         "     #include <(C standard library file)>\n"
         "     #include \"(user defined file)\"\n\n"
         "Do you want to see examples (type y for yes)?");
  examples = getchar();
  if(examples == 'y' || examples == 'Y') {
    printf("\n\n\n"
           "*** Include Examples ***\n\n"
           "     #include <stdio.h>\n"
           "This instance of include serves to add all the functions located\n"
           "in the standard library stdio.h file to the current source code.\n"
           "It will allow the code to use functions such as printf, and types\n"
           "such as FILE.\n\n"
           "     #include \"myheader.h\"\n"
           "In this instance, include takes all the contents from myheader.h,\n"
           "which is completely user defined.\n\n"
           "     #include \"module1.h\"\n"
           "Here, include is used to incorporate all the contents from a user\n"
           "defined module. This module can be part of a bigger project, where\n"
           "code separation and modularity become essential. Include allows\n"
           "developers to make their code as modular as they want.\n");
  }
}

#endif // !INCLUDE_H

```

```
*** Include ***

Include is a C preprocessor directive. It serves to incorporate
the contents of another file into the current one. Usually, it
is used to include header files from the C standard library or
user defined ones. Include is usually placed at the start of a
C file, before any of the other code. Included files can either
be in the same directory as the file that includes them or can
be added to the preprocessor using different flags depending
on the compiler. Common header files are
     stdio.h     // file and input/output functions and types
     stdlib.h    // memory allocation, random
     string.h    // see string function tutor
     stdbool.h   // boolean type
     math.h   // advanced math functions
Header files are text files, typically ending with the .h extension.
While they can contain full C code and definitions, they usually only
contain declarations. The point is to provide different source files
with public declarations for functions, variables, types, and more.
The actual definitions are usually in a corresponding c file. The C
code is then shared and used across all files that include the header,
without recompiling the definitions each time.

The syntax of include is
     #include <(C standard library file)>
     #include "(user defined file)"

Do you want to see examples (type y for yes)?y



*** Include Examples ***

     #include <stdio.h>
This instance of include serves to add all the functions located
in the standard library stdio.h file to the current source code.
It will allow the code to use functions such as printf, and types
such as FILE.

     #include "myheader.h"
In this instance, include takes all the contents from myheader.h,
which is completely user defined.

     #include "module1.h"
Here, include is used to incorporate all the contents from a user
defined module. This module can be part of a bigger project, where
code separation and modularity become essential. Include allows
developers to make their code as modular as they want.
```

mainin.h

```c
#ifndef MAININ_H
#define MAININ_H

void MainParameters(void) {
  char examples; // used for user input

  printf("*** Main Parameters ***\n\n"
         "Main functions in C can accept two different parameters. These\n"
         "are argc and argv. These are meant to be used for command-line\n"
         "arguments that are passed to the program when it is executed.\n"
         "The integer argc represents the amount of arguments (including\n"
         "the executable itself), while argv is an array of strings.\n"
         "Again, the name of the executable is always included, so argc\n"
         "will always be at least one, and argv at index 0 will be the\n"
         "executable's name.\n\n"
         "The syntax of main parameters is\n"
         "     int main(int argc, char *argv[]) {\n"
         "       // code\n"
         "       return 0;\n"
         "     }\n\n"
         "Use argc to get the amount of arguments.\n"
         "Use argv to get the arguments as strings.\n\n"
         "Do you want to see examples (type y for yes)?");
  examples = getchar();
  if(examples == 'y' || examples == 'Y') {
    printf("\n\n\n"
           "*** Main Parameters Examples ***\n\n"
           "     test.exe first second third\n"
           "This is an example of executing a C program from the command-line,\n"
           "while passing three arguments. In this case, argc would be equal\n"
           "to 4, while argv[0] would be test.exe, argv[1] would be first,\n"
           "argv[2] would be second, and argv[3] would be third.\n\n"
           "     printf(\"%%s\\n\", argv[0]);\n"
           "This code is printing the first argument in the argv parameter.\n"
           "This is going to be the executable name, so if the file is test.c\n"
           "and the executable is test.exe, this code will print ./test.exe.\n\n"
           "     printf(\"%%d, %%s\\n\", argc, argv[argc - 1]);\n"
           "In this code, n and the nth argument are being printed, where n\n"
           "is the amount of arguments (argc). Looking at the previous examples,\n"
           "the output would be '4, third'.\n\n"
           "     int i;\n"
           "     for(i = 0; i < argc; i++) {\n"
           "       printf(\"%%d, %%s\\n\", i, argv[i]);\n"
           "     }\n"
           "Here, each index and equivalent argument is being printed. Looking\n"
           "at the previous examples, the output would be '0, ./test', then '1,\n"
           "first', then '2, second', and then '3, third'.\n");
  }
}

#endif // !INCLUDE_H
```

```
*** Main Parameters ***

Main functions in C can accept two different parameters. These
are argc and argv. These are meant to be used for command-line
arguments that are passed to the program when it is executed.
The integer argc represents the amount of arguments (including
the executable itself), while argv is an array of strings.
Again, the name of the executable is always included, so argc
will always be at least one, and argv at index 0 will be the
executable's name.

The syntax of main parameters is
     int main(int argc, char *argv[]) {
       // code
       return 0;
     }

Use argc to get the amount of arguments.
Use argv to get the arguments as strings.

Do you want to see examples (type y for yes)?y



*** Main Parameters Examples ***

     test.exe first second third
This is an example of executing a C program from the command-line,
while passing three arguments. In this case, argc would be equal
to 4, while argv[0] would be test.exe, argv[1] would be first,
argv[2] would be second, and argv[3] would be third.

     printf("%s\n", argv[0]);
This code is printing the first argument in the argv parameter.
This is going to be the executable name, so if the file is test.c
and the executable is test.exe, this code will print ./test.exe.

     printf("%d, %s\n", argc, argv[argc - 1]);
In this code, n and the nth argument are being printed, where n
is the amount of arguments (argc). Looking at the previous examples,
the output would be '4, third'.

     int i;
     for(i = 0; i < argc; i++) {
       printf("%d, %s\n", i, argv[i]);
     }
Here, each index and equivalent argument is being printed. Looking
at the previous examples, the output would be '0, ./test', then '1,
first', then '2, second', and then '3, third'.
```