CS471: Program 4 – Parsing a Path

Write a C function that splits a full path into components. The input to your function should be a character string containing a path, e.g., `/u1/junk/cs471/sep11/readdisk.c`. Your function will return a list of character strings containing the components of the path. For the example, the return strings would be as follows.

0. `u1`
1. `junk`
2. `cs471`
3. `sep11`
4. `readdisk.c`
5. `NULL` – the NULL is important

Prototype your function as follows:

```
char **split(char *path);
```

Also add another function that frees the memory allocated. Prototype this function as follows.

```
void freelist(char **components);
```

In this case `components` is the list returned by `split`. If the path is incorrectly formed (could not possibly be a full path in any filesystem), return NULL. All memory for the component list should be dynamically allocated.

Create a subdirectory of your home directory named `sep20`, and save your functions there in a file named `split.c`. I will test your functions using the main program on the next page (there is a copy in the class junk directory). The `split.c` should contain only the two functions, not `main`. Due end of day, Friday September 20, 2019.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char **split(char *path);
void freelist(char **components);

int main(int argc, char *argv[])
{
   int i,j,k;
   char **args;

   for(i = 1 ; i < argc ; i++){
      args = split(argv[i]);
      if(!args)
         continue;
      for(j = 0 ; args[j] ; j++)
         printf("arg %d, component %d:  %s\n", i, j, args[j]);
      freelist(args);
   }
}
```