longestStrInAr

Write a C function `longestStrInAr()` that takes in an array of strings *`str`* and *`size`* (>0) as paramters, and returns the longest string and also the length of the longest string via the pointer parameter *`length`*. If two or more strings have the same longest string length, then the first appeared string will be retruned to the calling function. For example, if *`size`* is 5 and the array of strings is `{"peter","john","mary","jane","kenny"}`, then the longest string is `"peter"` and the string length is 5 will be returned to the calling function. The function prototype is:

```
char *longestStrInAr(char str[N][40], int size, int *length);
```

A sample C program to test the function is given below:

```
#include <stdio.h>
#include <string.h>
#define N 20
char *longestStrInAr(char str[N][40], int size, int *length);
int main()
{
   int i, size, length;
   char str[N][40], first[40], last[40], *p, *result;
   char dummychar;

   printf("Enter array size: \n");
   scanf("%d", &size);
   scanf("%c", &dummychar);
   for (i=0; i<size; i++) {
      printf("Enter string %d: \n", i+1);
      fgets(str[i], 40, stdin);
      if (p=strchr(str[i],'\n')) *p = '\0';
   }
   result = longestStrInAr(str, size, &length);
   printf("longest: %s \nlength: %d\n", result, length);
   return 0;
}
char *longestStrInAr(char str[N][40], int size, int *length)
{
   /* Write your code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Enter array size:
4
Enter string 1:
Kenny
Enter string 2:
Mary
Enter string 3:
Peter
Enter string 4:
Sun
longest: Kenny
length: 5
```

(2) Test Case 2:

```
Enter array size:
2
Enter string 1:
```

```
Sun
Enter string 2:
Mary
longest: Mary
length: 4
```