Problem 1.

Complete mystrcat() in the given source code.

Although you are not allowed to use library functions, you can add and use your own functions.

char *mystrcat(char *dest, char *src) : Appends a copy of the null-terminated byte string pointed to by *src* to the end of the null-terminated byte string pointed to by *dest*. The character *src[0]* replaces the null terminator at the end of *dest*. The resulting byte string is null-terminated.

- Parameters
  - dest - pointer to the null-terminated byte string to append to
  - src - pointer to the null-terminated byte string to copy from
- Return value
  - returns a copy of dest

```c
#include <stdio.h>

char *mystrcat(char *dest, char *src) {

  return dest;
}

int main(void) {
  char str[256] = "Hello ";
  char * pstr2 = "Good Bye World !!!";

  printf("%s\n", mystrcat(mystrcat(str, " World !!! ..."), pstr2 ));

  return 0;
}
```

Problem 2.

Complete print_all_permutations() in the given source code.

void print_all_permutations(char *str) : prints all permutations of a given string. For example, if the given string is ABC, the permutations are ABC, ACB, BAC, BCA, CBA, and CAB.

Print each permutation in a new line as below. (The output order can be different)

```
ABC
ACB
BAC
BCA
CBA
CAB
```

- Parameters
  - str - pointer to the null-terminated string. *You can assume that all characters in the string are different.*
- Return value - none

Test your program with test strings like ABC and ABCDE.

You can add new functions and variables.

You can use the C standard library functions if necessary.

```c
#include <stdio.h>

void print_all_permutations(char *str) {
  puts(str);
  return;
}

int main(void) {
  char str[256] = "ABC";

  print_all_permutations(str);
  return 0;
}
```

Problem 3.

Complete mystrtok1() in the given source code.

Although you are not allowed to use library functions, you can add and use your own functions.

char *mystrtok1(char *str, char delim) : Finds the next token in a null-terminated byte string pointed to by *str*. The separator character is identified by *delim*. This function is designed **to be called multiple times** to obtain successive tokens from the same string.

If *str* is not a null pointer, **the call is treated as the first call to strtok for this particular string**. The function searches for the first character which is not *delim*.

- If no such character was found, there are no tokens in *str* at all, and the function returns a null pointer.
- If such a character was found, it is the beginning of the token. The function then searches from that point on for the *delim*.
    - If *delim* character is not found, *str* has only one token, and future calls to strtok will return a null pointer
    - If *delim* is found, it is replaced by the null character '\0' and **the pointer to the following character is stored in a static location for subsequent invocations**.
- The function then returns the pointer to the beginning of the token

If *str* is a null pointer, **the call is treated as a subsequent call to strtok**: the function continues from where it was left in the previous invocation. The behavior is the same as if the previously stored pointer is passed as *str*.

- Parameters
    - str - pointer to the null-terminated byte string to tokenize
    - delim - a delimiter character for token separation
- Return value
    - Returns a pointer to the beginning of the next token or a null pointer if there are no more tokens.

```c
#include <stdio.h>

char *mystrtok1(char *str, char delim) {
  static char *psave = 0;

  return 0;
}

int main(void) {
  char pstr[256] = { ",123,hello,34 56, Good.,Bye"};
  char *ptoken;
  char delim = ',';
  int test = 0;

  ptoken = mystrtok1(pstr, delim);

  while (ptoken) {
    printf("%s\n",ptoken);
    ptoken = mystrtok1(0, delim);
  }

  return 0;
}
```

c

Problem 4.

Complete mystrtok(), **an extended one from the mystrtok1() in the previous problem**, in the given source code.

Although you are not allowed to use library functions, you can add and use your own functions.

char *mystrtok(char *str, char *delim) : Finds the next token in a null-terminated byte string pointed to by *str*. The separator characters are in a null-terminated byte string pointed to by *delim*. So we can use **multiple delimiter characters** in this version. This function is designed to be called multiple times to obtain successive tokens from the same string.

If str is not a null pointer, the call is treated as the first call to strtok for this particular string. The function searches for the first character which is not contained in *delim*.

- If no such character was found, there are no tokens in *str* at all, and the function returns a null pointer.
- If such a character was found, it is the beginning of the token. The function then searches from that point on for the first character **that is contained in *delim*.**
  - If no such character was found, *str* has only one token, and future calls to strtok will return a null pointer
  - If such a character was found, it is replaced by the null character '\0', and the pointer to the following character is stored in a static location for subsequent invocations.
- The function then returns the pointer to the beginning of the token

If str is a null pointer, the call is treated as a subsequent call to strtok: the function continues from where it was left in the previous invocation. The behavior is the same as if the previously stored pointer is passed as *str*.

- Parameters
  - str - pointer to the null-terminated byte string to tokenize
  - delim - pointer to the null-terminated byte string identifying delimiters
- Return value
  - Returns a pointer to the beginning of the next token or a null pointer if there are no more tokens.