Function Parameter Passing and Return Assignments

1. Refer the code below and find the issue.

```
#include<stdio.h>
int *func(void);
int main()
{
int num,*ptr = NULL;
ptr = (int *)func();
num = *ptr;
return 1;
}
int *func()
{
int local;
local = 10;
return(&local);
}
```

In above code is there a way(s) to return local variable address to caller?

Sol:

```
#include <stdio.h>

int *func(void);


int main() {
   int num, *ptr = NULL;
   ptr = func();
   num = *ptr;
   printf("Value: %d\n", num);
   return 0;
}
int *func() {
   static int local = 10;
   return &local;
}
```

Reason:

The local variable in func() is declared as static, so it **persists** even after the function ends. The pointer returned by func() will always point to this valid memory location.

2. Write a program with a function read_extract_characters() to read a string (of max length 50 characters) from user, extract the characters at odd indices, store in an other array and return to the call. Caller should be able to read and display the extracted string.
[Note : do not return a local variable in function to caller]

Sol:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


char* read_extract_characters(void);

int main() {

   char *extractedString;

   extractedString = read_extract_characters();

   printf("Extracted characters at odd indices: %s\n", extractedString);

   free(extractedString);

return 0;

}


char* read_extract_characters(void) {

   char input[51];

   int length, extractedIndex = 0;

   printf("Enter a string (max 50 characters): ");

   fgets(input, sizeof(input), stdin);


   input[strcspn(input, "\n")] = '\0';

   length = strlen(input);
```

```
    char *extracted = (char *)malloc((length / 2 + 1) * sizeof(char));
    for (int i = 1; i < length; i += 2) {
        extracted[extractedIndex++] = input[i];
    }
    extracted[extractedIndex] = '\0';
    return extracted;
}
```

Output:

```
user57@trainux01:~/Batch17OCT2024/function$ vi function2.c
user57@trainux01:~/Batch17OCT2024/function$ gcc function2.c
user57@trainux01:~/Batch17OCT2024/function$ ./a.out
Enter a string (max 50 characters): iswaryapothala
Extracted characters at odd indices: sayptaa
```

3 Write below functions to extract and return the required inputs from an email id string of max length 80 characters. Program should be able to detect an invalid email id too based on following validations. Also value returned should be in scope and accessible in caller.

a. valid email id will have username, host and domain name (as .com/.edu)

Functions:

char *getuser(char input[]); // return NULL or valid username from email id input

char *gethost(char input[]); //return NULL or valid hostname from email id input

char *getdomain(char input[]); //return NULL or valid domain name from email id input

int isValidDomain(char input[]); // return 1 if domain is “.com” or “.edu” else 0

sol:

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

```
int isLetter(char ch) {
   return ('a' <= ch && ch <= 'z') || ('A' <= ch && ch <= 'Z');
}

int isValidDomain(char input[]) {
   char *dotPos = strrchr(input, '.');
   if (dotPos != NULL) {
      if (strcmp(dotPos, ".com") == 0 || strcmp(dotPos, ".edu") == 0) {
         return 1;
      }
   }
   return 0;
}

char* getuser(char input[]) {
   char *userEnd = strchr(input, '@');
   if (userEnd != NULL) {
      size_t len = userEnd - input;
      char *username = (char*)malloc(len + 1);
      if (username != NULL) {
         strncpy(username, input, len);
         username[len] = '\0';
         return username;
      }
   }
   return NULL;
}

char* gethost(char input[]) {
   char *userEnd = strchr(input, '@');  // Find '@'
   if (userEnd != NULL) {
```

```
        char *hostStart = userEnd + 1;
        char *dotPos = strchr(hostStart, '.');
        if (dotPos != NULL) {
            size_t len = dotPos - hostStart;
            char *hostname = (char*)malloc(len + 1);
            if (hostname != NULL) {
                strncpy(hostname, hostStart, len);
                hostname[len] = '\0';
                return hostname;
            }
        }
    }
    return NULL;
}

char* getdomain(char input[]) {
    char *dotPos = strrchr(input, '.');
    if (dotPos != NULL && isValidDomain(input)) {
        char *domain = strdup(dotPos + 1);
        return domain;
    }
    return NULL;
}
int main() {
    char email[81];
    printf("Enter email ID: ");
    fgets(email, sizeof(email), stdin);
    email[strcspn(email, "\n")] = '\0';
    if (strchr(email, '@') == NULL || strchr(strrchr(email, '@'), '.') == NULL) {
        printf("Invalid email format.\n");
        return 1;
```

```
    }
    char *user = getuser(email);
    char *host = gethost(email);
    char *domain = getdomain(email);

    if (user != NULL && host != NULL && domain != NULL && isValidDomain(email)) {
        printf("Username: %s\n", user);
        printf("Host: %s\n", host);
        printf("Domain: %s\n", domain);
    } else {
        printf("Invalid email ID.\n");
    }
    if (user != NULL) free(user);
    if (host != NULL) free(host);
    if (domain != NULL) free(domain);

    return 0;
}
```

Output: