Answers:

Exercise: 1

1.compile with flag -g,then load executable using gdb 'executable',then "run"
2.run <arglist>
3.b <line number>
4.break <line number> if (expr)
5.s
6.s or n
7.c
8.print <variable_name>
9.display
10.‘i lo’ - info locals:values of all local variables in current function
11.quit or q

Exercise: 2

1.I have HIGHLIGHTED the code which I modified.
2.No,they are not correct.
3.previous values and junk values are also shown.
4.Now to fix this error, we should not print the extra values apart from the str1 and str2, so we put a terminal symbol '\0' after the length of(str1+str2).
Below is the code for the above explanation.

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

/*
   Return the result of appending the characters in s2 to s1.
   Assumption: enough space has been allocated for s1 to store the extra
   characters.
*/
char* append (char s1[ ], char s2[ ]) {
   int s1len = strlen (s1);
   int s2len = strlen (s2);
   int k;
   for (k=0; k<s2len; k++) {
      s1[k+s1len] = s2[k];
   }
   s1[s1len+s2len] = '\0';
   return s1;
}

int main ( ) {
   char str1[10];
   char str2[10];
   while (1) {
      printf ("str1 = ");
      if (!gets (str1)) {
         return 0;
```

```
        };
        printf ("str2 = ");
        if (!gets (str2)) {
            return 0;
        };
        printf ("The result of appending str2 to str1 is %s.\n",
            append (str1, str2));
    }
    return 0;
}
```

Exercise 3:

```
 1 #include <stdio.h>
 2
 3 /*
 4    Read a set of values from the user.
 5    Store the sum in the sum variable and return the number of values read.
 6 */
 7 int read_values(double *sum)
 8 {
 9   int values=0,input=0;
10   *sum = 0;
11   printf("Enter input values (enter 0 to finish):\n");
12   scanf("%d",&input);
13   while(input != 0) {
14     values++;
15     *sum += input;
16     scanf("%d",&input);
17   }
18   return values;
19 }
20
21 int main()
22 {
23   double sum=0;
24   int values;
25   values = read_values(&sum);
26   printf("Average: %g\n",sum/values);
27   return 0;
28 }
```

LS1B:

PART A:
NO bug is present.

PART B:
YES , Memory Leak.

PART C:
NO.

LS1C:

Valgrind-test.c :

Helped us to understand about the valgrind profiler much better.

debug-test.c:

Segmentation fault occured because of the NULL pointer.
Below is the modified code:

```
 1 #include "stdio.h"
 2
 3 void
 4 print_scrambled(char *message)
 5 {
 6   int i = 3;
 7   if(message)
 8   {
 9     do
10     {
11       printf("%c", (*message)+i);
12     } while (*++message);
13   }
14   printf("\n");
15 }
16
17 int
18 main()
19 {
20   char * bad_message = NULL;
21   char * good_message = "Hello, world.";
22
23   print_scrambled(good_message);
24   print_scrambled(bad_message);
25 }
```

OUTPUT
After running this modified program, output is as below:
Khoor/#zruog1