Write a C program that inputs a line of text from standard input, and prints the line to standard output in reverse order. Write the program to handle an input line that has at most 100 characters. If the input line has more than 100 characters, print an error message to standard error and exit without printing anything to standard output. Use the `fscanf` function to read the input text from standard input.
Write your solution to Question (i) on this page. Please clearly indicate your solution and show all work on this page.
(ii)  (5 pts.)

Consider the following C program.

```c
#include <stdio.h>
#define OFFSET 2
#define SIZE 5

int f(int *x, int y) {
    y = (*x) + 3;
    y++;
    (*x) = y - 10;
    x++;
    return (*x) + y;
}

int main(void) {
    int a[SIZE] = {3, -5, 7, 1, 9};
    int *z = NULL;
    int result = 0;
    int i = 0;
    
    z = &a[2];
    result = f(z, a[0]);
    printf("r1 = %d\n", result);
    result = f(z, *z);
    printf("r2 = %d\n", result);
    return 0;
}
```

Part 1: Show the complete output as it appears on standard output.

Part 2: Show the contents of the array \(a\) (all five values in the proper sequence) immediately after the first call to function \(f\).

Part 3: Show the contents of the array \(a\) (all five values in the proper sequence) immediately after the second call to function \(f\).
Write your solution to Question (ii) on this page. Please clearly indicate your solution and show all work on this page.
(iii)  (5 pts.)

Consider the following C program, which includes three function definitions, including the main function shown on the next page.

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

#define LIMIT 3
#define DISPLAY_LENGTH 8

struct element {
    char text;
    struct element *next;
};

struct element *add_elements(struct element *h, char c1, char c2) {
    struct element *t1 = NULL;
    struct element *t2 = NULL;
    t1 = malloc(sizeof(struct element));
    t2 = malloc(sizeof(struct element));
    t1->text = c1;
    t2->text = c2;
    t1->next = t2;
    if (h == NULL) {
        t2->next = t1;
    } else {
        t2->next = h->next;
        h->next = t1;
    }
    return t1;
}

void print_elements(struct element *elements, int num) {
    int i = 0;
    struct element *p = NULL;
    p = elements;
    for (i = 0; i < num; i++) {
        printf("item %d: %c\n", i, p->text);
        p = p->next;
    }
}
```
int main(void) {
    struct element *data = NULL;
    char *name1 = "mickey";
    char *name2 = "donald";
    int i = 0;

    for (i = 0; i < LIMIT; i++) {
        data = add_elements(data, name1[i], name2[i]);
    }
    print_elements(data, DISPLAY_LENGTH);
    return 0;
}

Show the output of this program exactly as it appears on standard output. Show your work.
Write your solution to Question (iii) on this page. Please clearly indicate your solution and show all work on this page.
Consider the following function prototype and the associated header comment.

```c
/* Return the total number of bytes used to store the string pointed to by s. */
int string_size(char *s);
```

Part 1: Develop a complete C code implementation of the function `string_size`. Your implementation should NOT include calls to any functions or macros, not even functions or macros from the standard C library.

Part 2: Develop a C program that performs the following tasks (in the order given):

A. Declares a string constant that has the text "hello" (excluding the double quotes).

B. Calls `string_size` to determine the number of bytes in this string constant.

C. Prints the result as a decimal number to standard output.

Part 3: For the C program specified in Part 2, show the output of the program exactly as it appears on standard output.
Write your solution to Question (iv) on this page. Please clearly indicate your solution and show all work on this page.
Problem 1:

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

#define LIMIT 100

int main(void) {
    char line[LIMIT];
    int i = 0;
    int done = 0;
    char c = '\0';

    while (!done) {
        fscanf(stdin, "%c", &c);
        if (c == EOF) {
            done = 1;
        } else if (c == '\n') {
            done = 1;
        } else if (i == LIMIT) {
            fprintf(stderr, "Line too long");
            exit(1);
        } else {
            line[i++] = c;
        }
    }

    while (i >= 0) {
        printf("%c", line[i--]);
    }

    printf("\n");

    return 1;
}
```
Problem 2:

Part 1:

\[ r_1 = 12 \]
\[ r_2 = 6 \]

Part 2:

Array values: 3, -5, 1, 1, 9

Part 3:

Array values: 3, -5, -5, 1, 9

Problem 3:

item 0: c
item 1: n
item 2: o
item 3: d
item 4: m
item 5: i
item 6: c
item 7: n
Problem 4:

Part 1:

```c
#include <stdio.h>

int string_size(char *s) {
    int i = 0;
    int length = 0;

    for (i = 0; s[i] != '\0'; i++) {
        length += sizeof(char);
    }

    /* include the terminating null character */
    length += sizeof(char);

    return length;
}
```

Part 2:

```c
int main(void) {
    const char *s = "hello";
    int result = 0;

    result = string_size(s);

    printf("The result is: %d\n", result);
    return 0;
}
```

Part 3:

The result is: 6