

Documentation of Day 14

Exercise 4-9

Question:

Our getch and ungetch do not handle a pushed-back EOF correctly. Decide what their properties ought to be if an EOF is pushed back, then implement your design.

Answer:

The problem is distinguishing the end of input from valid data. The solution is that getchar returns a distinctive value when there is no more input, a value that cannot be confused with any real character. This value is called EOF, for "end of file". We must declare c to be a type big enough to hold any value that getchar returns. We can't use char since c must be big enough to hold EOF in addition to any possible char. Therefore we use int.

Source Code:

```
#include <stdio.h>

#define BUFSIZE 100

int buf[BUFSIZE]; // Buffer to hold characters
int bufp = 0;

int getch(void) {
    return (bufp > 0) ? buf[--bufp] : getchar();
}

void ungetch(int c) {
    if (bufp >= BUFSIZE) {
        printf("ungetch: too many characters\n");
    } else {
        buf[bufp++] = c;
    }
}

int main() {
    int c;
```

```

// Test case 1: Pushing back 'A' and 'B'
ungetch('A');
ungetch('B');

// Reading characters
while ((c = getch()) != EOF) {
    printf("%c", c);
}
putchar('\n');

// Test case 2: Pushing back EOF
ungetch EOF);
    printf("%d", c);
// Reading characters
while ((c = getch()) != EOF) {
    printf("%c", c);
}
putchar('\n');
return 0;
}

```

- This code includes the definitions of the `getch`, `ungetch`, and `main` functions. It also includes the buffer `buf` and the buffer pointer `bufp`.
- The `getch` function retrieves characters from the buffer if there are any available. If the buffer is empty, it reads characters directly from the standard input using `getchar`. It correctly handles the EOF character, returning it when encountered.
- The `ungetch` function stores characters in the buffer unless the buffer is full (`bufp >= BUFSIZE`). It can handle the EOF character as well.
- In the `main` function, there are two test cases. The first test case pushes back the characters 'H' and 'e', which are then retrieved and printed. The second test case pushes back the EOF character, which terminates the loop when encountered.

Exercise 4-13

Question:

Write a recursive version of the function reverse(s), which reverses the string s in place.

Source Code:

```
#include <stdio.h>

#include<string.h>

#define SIZE 100

void reverse(char s1[], int index, int size) {

    char temp;

    temp = s1[index];

    s1[index] = s1[size - index];

    s1[size - index] = temp;

    if (index == size / 2) {

        return;

    }

    reverse(s1, index + 1, size);

}

int main() {

    char str[SIZE];

    printf("Enter a string to reverse: ");

    fgets(str, sizeof(str), stdin);

    reverse(str, 0, strlen(str) - 1);

    printf("The string after reverse is: %s", str);

    return 0;

}
```

The Reverse Function: The reverse function is a recursive function that calls itself. It accepts three parameters which are:

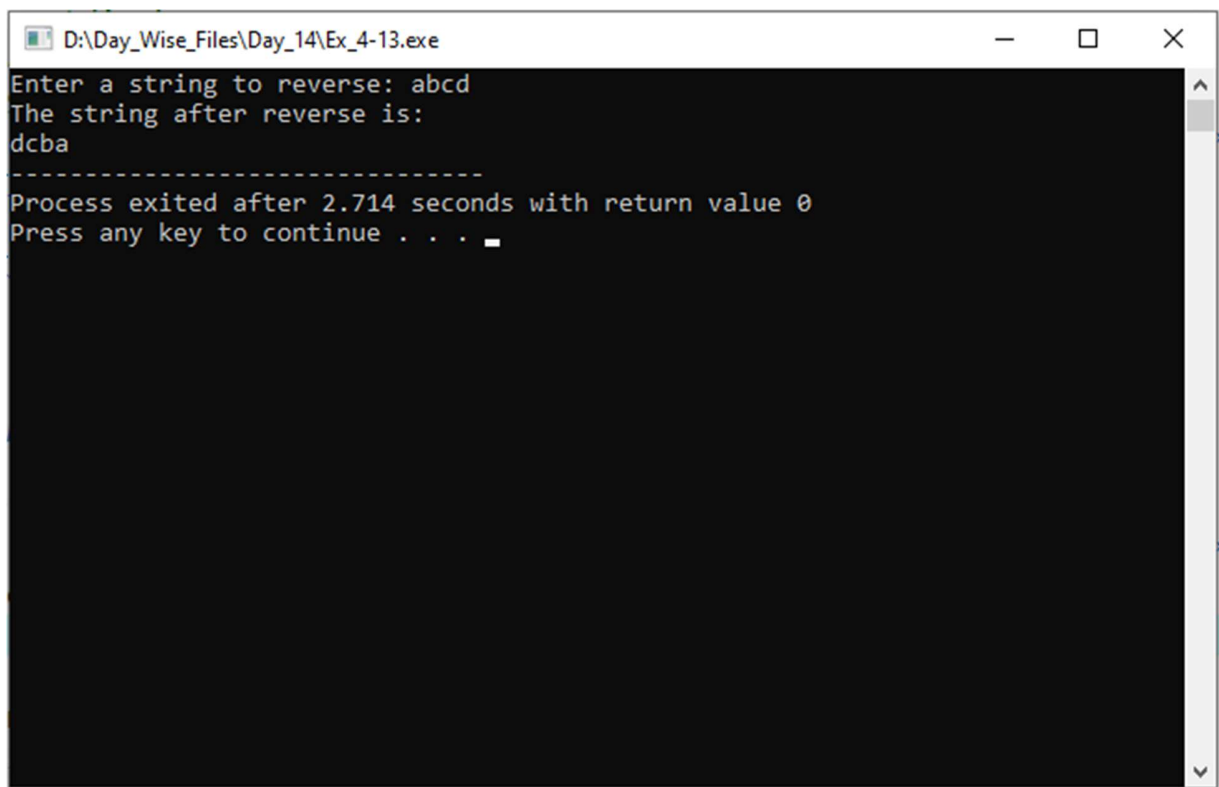
s1[] – The string to reverse.

index – The starting index

size – Size of the string

The reverse function swaps the first element of the string with the element at the (size – index)'th index of the string. Then checks if the current index of the value that is being swapped is equal to the half of the size (size/2) of the whole string. If the condition is true that means the swapping of all elements is done and the string is reversed fully. This is the base case where the recursion stops. At last the function reverse is invoked with the parameters, but this time the index is incremented by 1 as the last value is swapped. After the whole recursion process is done the string is reversed and printed from the main() function.

Output:



```
D:\Day_Wise_Files\Day_14\Ex_4-13.exe
Enter a string to reverse: abcd
The string after reverse is:
dcba
-----
Process exited after 2.714 seconds with return value 0
Press any key to continue . . .
```

Exercise 4-14:

Question:

Define a macro swap(t,x,y) that interchanges two arguments of type t. (Block structure will help.)

Source Code:

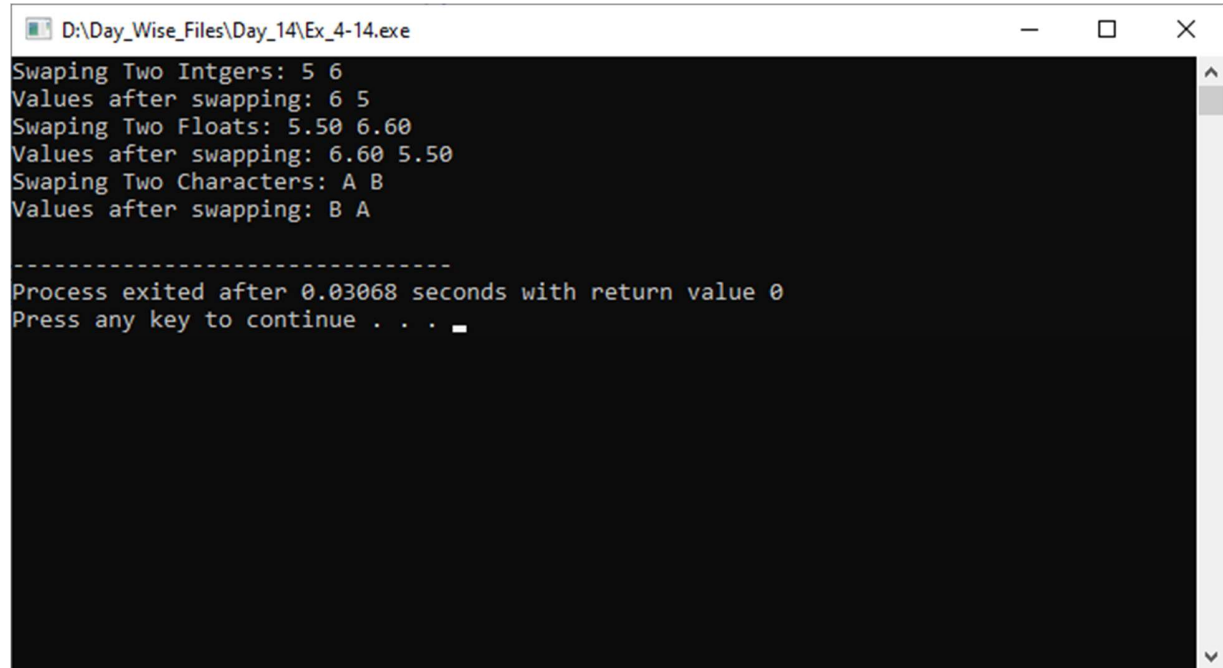
```
#include <stdio.h>
#include <string.h>
#define swap(type, x, y) {type temp = x; x=y; y=temp;}

int main() {
    int a = 5, b = 6;
    printf("Swaping Two Intgers: %d %d\n", a, b);
    swap(int, a, b);
    printf("Values after swapping: %d %d\n", a, b);

    float x = 5.5, y = 6.6;
    printf("Swaping Two Floats: %f %f\n", x, y);
    swap(float, x, y);
    printf("Values after swapping: %f %f\n", x, y);

    char p = 'A', q = 'B';
    printf("Swaping Two Characters: %c %c\n", p, q);
    swap(char, p, q);
    printf("Values after swapping: %c %c\n", p, q);
    return 0;
}
```

Output:



```
D:\Day_Wise_Files\Day_14\Ex_4-14.exe
Swaping Two Intgers: 5 6
Values after swapping: 6 5
Swaping Two Floats: 5.50 6.60
Values after swapping: 6.60 5.50
Swaping Two Characters: A B
Values after swapping: B A

-----
Process exited after 0.03068 seconds with return value 0
Press any key to continue . . .
```