Assignment 4

Linux and C Programming (62558)

Mads Richardt (s224948)

December 5, 2022

Contents

Feedback

7.1 Correct answer. The only optimization would be to use the condition in the while loop to avoid the if construct.

7.2 Correct and the same as before with the while loop.

7.3 Good solution and again maybe give your while condition a extra look so you don't use a magic number without a comment.

Annotation Notes on PDF Submission

1. Should have been it own function but this clearly shows that you understand how to swap two variables.
2. Correct and very good answer.

Exercise 7

Original Submission

```
/*
Mandatory assignment: 4
Lesson: 7
```

```c
/*
Student name: Mads Richardt
Student Id: s224948
Date: 20/10/2022
*/

#include <time.h>
#include <stdlib.h>
#include <stdio.h>

int main(void) {

    srand(time(0));
    int guess;
    int guess_counter = 0;
    int select_var;
    int random_number;

    // Welcome message
    puts("**********");
    puts("Exercise 7");
    puts("**********");
    puts("");

    // Select exercise
    printf("1: SizeOfInt\n2: GuessANumber\n3: GuessANumberReverse\nPlease select option: ");
    scanf("%d", &select_var);

    // SizeOfInt
    if (select_var == 1) {

        puts("");
        puts("************");
        puts("SizeOfInt");
        puts("************");

        int counter = 0;
        int n_new = 0;
        int n_old = 1;
        while (1) {
            n_new = 2 * n_old;
            counter++;
            if (n_new == n_old) {
                counter--;
                break;
            }
        n_old = n_new;
        }

        printf("Number of bits in int: %d\n", counter);
    }

    // GuessAnumber
```

```
if (select_var == 2) {


    random_number = rand()%100 + 1;
    puts("");
    puts("************");
    puts("GuessANumber");
    puts("************");

    while(1) {
        guess_counter++;
        printf("Guess number: ");
        scanf("%d", &guess);

        if (guess == random_number) {
            puts("Correct!!!");
            printf("You guessed the secrete number on guess %d\n", guess_counter);
            break;
        }

        if (guess > random_number) {
            puts("Guess is larger than the secrete number");
        }
        else {
            puts("Guess is lower than the secrete number");
        }
    }
}

// GuessANumberReverse
if (select_var == 3) {
    int HIGH = 100;
    int LOW = 0;
    puts("");
    puts("******************");
    puts("GuessANumberReverse");
    puts("******************");
    puts("");

    // Get number fro user
    printf("Enter number: ");
    scanf("%d", &random_number);

    while(guess_counter < 20) {
        guess = LOW + (HIGH - LOW)/2;
        guess_counter++;
        printf("Guess: %d\n", guess);
        //printf("Low: %d\n",LOW);
        //printf("High: %d\n",HIGH);
        //puts("");
        if (random_number == guess) {
            printf("I guessed your number on guess %d\n", guess_counter);
```

```
                break;
            }
            if (guess > random_number) {
                HIGH = guess;
            }
            else {
                LOW = guess;
            }
        }
    }

    // Program  terminated without error
    return 0;
}
```

Updates

In the updated listed below, I have added conditions to the while loops.

```
/*
Mandatory assignment: 4
Lesson: 7
Student name: Mads Richardt
Student Id: s224948
Date: 20/10/2022
*/

#include <time.h>
#include <stdlib.h>
#include <stdio.h>

int main(void) {

    srand(time(0));
    int guess;
    int counter;
    int select_var;
    int random_number;
    int oldInt = 0;
    int newInt = 1;
    int HIGH = 101;
    int LOW = 0;

    // Welcome message
    puts("**********");
    puts("Exercise 7");
    puts("**********\n");

    // Select exercise
    printf("1: SizeOfInt\n2: GuessANumber\n3: GuessANumberReverse\nPlease select option: ");
    scanf("%d", &select_var);
```

```
// SizeOfInt
if (select_var == 1) {
    puts("\n*********");
    puts("SizeOfInt");
    puts("*********");
    counter = 1;

    while (newInt != oldInt) {
        oldInt = newInt;
        newInt = 2*oldInt;
    }

    printf("Number of bits in int: %d\n", counter);
}

// GuessAnumber
if (select_var == 2) {
    puts("\n************");
    puts("GuessANumber");
    puts("************");
    random_number = rand()%100 + 1;
    counter = 0;

    while (guess != random_number) {
        counter++;
        printf("Guess number: ");
        scanf("%d", &guess);

        if (guess > random_number) {
            puts("Guess is larger than the secrete number");
        }
        else {
            puts("Guess is lower than the secrete number");
        }
    }
    printf("Correct!!!\nYou guessed the secrete number on guess %d\n", counter);
}

// GuessANumberReverse
if (select_var == 3) {
    puts("\n******************");
    puts("GuessANumberReverse");
    puts("******************\n");
    counter = 0;
    printf("Enter number in the range [%d - %d]: ", LOW, HIGH-1); // Get number from user
    scanf("%d", &random_number);

    while (random_number != guess) {
        guess = LOW + (HIGH - LOW)/2;
        counter++;
        printf("Guess %d: %d\n",counter, guess);
```

```
            if (guess > random_number) {
                HIGH = guess;
            }
            else {
                LOW = guess;
            }
        }
    }

    return 0; // Program  terminated without error
}
```

Exercise 8

Original Submission

Exercise 8.2

```
Stack
-----
int x = 1
int y = 2
int temp = *y;
*y = *x;
*x = temp;
```

Exercise 8.3

When the StackAllocation function is called we get the following output.

```
***************
StackAllocation
***************
Value of x = 7
Address of x = 0x7ffc7811b05c
Value of y = 7
Address of y = 0x7ffc7811b05c
```

When foo1 is called, the variable x gets declared, meaning that x is allocated the address at the top of the stack. Then, x gets initialized to 7, meaning that 7 gets stored at the address allocated to x. When foo1 exits, the address assigned to x is freed up. Subsequently, when foo2 is called the address, which was formerly allocated to x, is now allocated to y. However, y is not initialized to anything, so the address still stores the value assigned to x in foo1. Accordingly, the addresses and values of x and y are identical.

Updates

Unfortunately, I forgot to append the source code in the original submission - it can be found in the following section.

Source Code

```
/*
Mandatory assignment: 4
```

```c
/*
Lesson: 8
Student name: Mads Richardt
Student Id: s224948
Date: 20/10/2022

************
Exercise 8.1
************
    Answer:
        D. *p = 75
*/

#include <stdio.h>

void swap(int *,int *);
void foo1(int);
void foo2(int);

int main() {
    int select_var;

    // Select exercise
    printf("1: Swap\n2: StackAllocation\nPlease select option: ");
    scanf("%d", &select_var);

    // Exercise 8.2
    if (select_var == 1) {
        puts("\n****");
        puts("Swap");
        puts("****");
        int x = 1;
        int y = 2;
        puts("Variable values before swap was called");
        printf("x = %d, y = %d\n",x,y);
        // Call swap
        swap( &x, &y);
        puts("Variable values after swap was called");
        printf("x = %d, y = %d\n",x,y);
    }

    if (select_var == 2) {
        puts("\n***************");
        puts("StackAllocation");
        puts("***************");
        foo1(7);
        foo2(11);
    }
    return (0);
}

void swap(int *x, int *y) {
    int temp = *y;
```

```
    *y = *x;
    *x = temp;
}

void foo1 (int xval) {
    int x;
    x = xval;
    printf("Value of x = %d\nAddress of x = %p\n", x, &x);
}

void foo2(int dummy) {
    int y;
    printf("Value of y = %d\nAddress of y = %p\n", y, &y);
}
```