भारतीय सूचना प्रौद्योगिकी संस्थान गुवाहाटी

Indian Institute of Information Technology Guwahati

COMPUTER PROGRAMMING LAB (CS110)

ASSIGNMENTS–07

1. Write a program in C to read the following types of variables using the `scanf` function: `char`, `int`, `float`, and `double`.

2. write a function in C that takes two pointers to integers and swaps the variables.
   Solution:

```c
#include <stdio.h>

int main() {
    int a = 2, b = 3;
    void swap(int *, int *);
    swap(&a, &b);
    printf("a = %d, b = %d", a, b);
}

void swap(int *pa, int *pb) {
    *pa ^= *pb;
    *pb ^= *pa;
    *pa ^= *pb;
}
```

3. Write a program in C to create an array of type `int` of size 5. Use the `scanf` function to take user inputs initializing the array. Then, print the elements of the array.
   Solution 1:

```c
#include <stdio.h>

int main() {
    int array[5];
    for (int i = 0; i < 5; i ++) {
        scanf("%d", array + i);
    }
    for (int i = 0; i < 5; i ++) {
        printf("%d ", *(array + i));
    }
}
```

Solution 2:

```c
#include <stdio.h>

int main() {
    int array[5];
    for (int i = 0; i < 5; i ++) {
        scanf("%d", &array[i]);
    }
    for (int i = 0; i < 5; i ++) {
        printf("%d ", array[i]);
    }
}
```

4. Write a program in C to dynamically alllocate an array of type `int`. The size of the array is an user input. Use the `scanf` function to take user inputs initializing the array. Then, print the elements of the array. Reuse the same memory to store an array of `char`. Now, free the allocated memory from the heap.

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

int main() {
    int size = 0, *array = NULL;
    printf("Enter the size of the array: ");
    scanf("%d", &size);
    array = (void *) calloc(size, sizeof(int));
    //array = (void *) malloc(size * sizeof(int)); // An alternative of previous line
    printf("Enter the elements of the int-array (size is %d): ", size);
    for (int i = 0; i < size; i ++) {
        scanf("%d", array + i);
    }
    printf("The elements of the int-array: ");
    for (int i = 0; i < size; i ++) {
        printf("%d ", *(array + i));
    }

    char *c_array = (char *) array; // reusing the same memory
    printf(
        "\nEnter the elements of the char-array (size is %d): ",
        ((int) (size * sizeof(int)))
    );
    for (int i = 0; i < size * sizeof(int); i ++) {
        scanf("%c", &c_array[i]);
    }
    printf("The elements of the char-array: ");
    for (int i = 0; i < size * sizeof(int); i ++) {
        printf("%c ", c_array[i]);
    }

```

```
    printf("\nThe modified elements of the int-array: ");
    for (int i = 0; i < size; i ++) {
        printf("%d ", array[i]);
    }

    free(array);
    array = NULL; // handling dangling/ wild pointer
    c_array = NULL; // handling dangling/ wild pointer
}
```

5. Realize the following multi-file program:

   The following code is stored in a file named `my_library.h`:

```
#include <stdio.h>

extern int global_1; // declaration; cannot initialize

void print_static_global();
extern void print_global(); // extern does not have any effect

/* If included, the following will cause error */
//void sayHello();

void sayHi(); // This is never defined
```

   The following code is stored in a file named `my_library.c`:

```
/* Use the following to compile:
gcc -c my_library.c
*/
#include "my_library.h"

/* Can be accessed from anywhere */
int global_1 = 1; // definition; this is good
int global_2 = 2; // multiple-definition; undefined behavior
int global_3 = 3; //

/* Can be accessed from anywhere */
extern void print_global() { //the keyword extern does not have any effect
    printf("\n*** File: %s, Function: %s, Line: %d ***\n",
        __FILE__, __func__, __LINE__
    );
    printf(
        "global_1 = %d, &global_1 = %p\n",
        global_1, &global_1
    );
    printf(
        "global_2 = %d, &global_2 = %p\n",
        global_2, &global_2
    );
```

```
    printf(
        "global_3 = %d, &global_3 = %p\n",
        global_3, &global_3
    );
    printf("\n*** File: %s, Function: %s, Line: %d ***\n",
        __FILE__, __func__, __LINE__
    );
}

/* Access restricted to this file only */
static int static_global_1 = 4;
static int static_global_2 = 5;

/* Can be accessed from anywhere */
void print_static_global() {
    printf("\n*** File: %s, Function: %s, Line: %d ***\n",
        __FILE__, __func__, __LINE__
    );
    printf(
        "static_global_1 = %d, &static_global_1 = %p\n",
        static_global_1, &static_global_1
    );
    printf(
        "static_global_2 = %d, &static_global_2 = %p\n",
        static_global_2, &static_global_2
    );
    printf("\n*** File: %s, Function: %s, Line: %d ***\n",
        __FILE__, __func__, __LINE__
    );
}

/* Access restricted to this file only */
static void sayHello() {
    printf("Hello\n");
};
```

The following code is stored in a file named `main.c`:

```
/* Use the following to compile and link:
gcc -c main.c a.c -o executable.exe
*/

#include "my_library.h"

//int global_1; //Need not to do it; already in my_library.h
int global_2; // multiple-definition; undefined behavior
// no declaration or definition of global_3

static int static_global_1 = 100;

int main() {
```

```
    /*First, we look for global variables*/
    printf(
        "global_1 = %d, &global_1 = %p\n",
        global_1, &global_1
    );
    /*The following will cause undefined behavior.
    This is because global_2 is defined twice. */
    printf(
        "global_2 = %d, &global_2 = %p\n",
        global_2, &global_2
    );
    /*If included, the following will cause an error.
    This is because global_3 is neither declared nor
    defined. */
    //printf(
    //    "global_3 = %d, &global_3 = %p\n",
    //    global_3, &global_3
    //);

    print_global();


    /* Now, we look for static*/
    printf(
        "static_global_1 = %d, &static_global_1 = %p\n",
        static_global_1, &static_global_1
    );
    /*If included, the following will cause an error */
    //printf(
    //    "static_global_2 = %d, &static_global_2 = %p\n",
    //    static_global_2, &static_global_2
    //);

    print_static_global();

    /* If included, this will cause an error as it is undefined. */
    //sayHello();

    /* If included, this will cause an error as it is undefined. */
    //sayHi();

    return 0;
}
```