Lab 7

You are expected to copy and paste your code into each corresponding box in this handout and submit it as a Word document or PDF file before the due. Additionally, your lab instructor will tell you which three questions you must showcase during the lab session. While you may demonstrate your code running in person after the due date, your file must be submitted on time.

Hint #1: For questions asking for 'a **function** which takes in xyz', the xyz refer to function parameters, not the use of scanf().

Hint #2: For questions asking for 'a **program** which reads in xyz', the xyz refer to the use of scanf().

Hint #3: For questions asking for 'a program', you are expected to include all necessary #include and define your main function. It's up to you whether to define additional helper functions, though usually, you don't need to.

Task 1: To define a **program** reads in an integer (in decimal) from user, and then prints it in octal, decimal and hex.

```
#include <stdio.h>

int main(void) {
     int input = 0;
     printf("Enter a decimal integer: ");
     scanf("%d", &input);
     printf("%#o\n%d\n%#x\n", input, input, input);
     return 0;
}
```

Task 2: To define a **program** reads in a float number from user, and then prints it with "%g", "%f" and "%e".

```
#include <stdio.h>

int main(void) {
    float input = 0;
    printf("Enter a float: ");
    scanf("%f", &input);
    printf("%g\n%f\n%e\n", input, input, input);
    return 0;
}
```

Task 3: To define a **program** reads in a float number from user, and then prints the equivalent percentage.
For example, input: 0.123 -> print 12.3%

```
#include <stdio.h>

int main(void) {
     float input = 0;
     printf("Enter a float: ");
     scanf("%f", &input);
     printf("%.2f%%\n", input * 100);
     return 0;
}
```

Task 4: To define a **program** to read in three numbers, and then to print them all twice with in two separate lines as below:
1) Frist time print, to make each of the numbers takes no less than 10 char spaces
2) Second time print, to align to left and each of them takes no less then 6 char spaces

Example: if input is 1, -200, 0

Output:
```
         1       -200          0
1   -200  0
```

```
#include <stdio.h>

int main(void) {
     int input_0, input_1, input_2;
     printf("Enter three numbers: ");
     scanf("%d %d %d", &input_0, &input_1, &input_2);
     printf("%10d %10d %10d\n", input_0, input_1, input_2);
     printf("%-6d %-6d %-6d\n", input_0, input_1, input_2);
     return 0;
}
```

Task 5: To define a **program** reads in three positive numbers (one integer in decimal, one integer in hex, and the last one as float in decimal), and then print them all three times (in each line of output, use comma to separate each number)

1) Print them all in octal (for the last number, only consider the integer part)
2) Print them all in hex with prefix "0x"(for the last number, same as above)
3) Print them all in float in decimal with at least 2 decimal digits

Eg: input as 1 11 11.11
Output:
1,21,13
0x1,0x11, 0xb
1.00, 17.00, 11.11

```
#include <stdio.h>

int main(void) {
     int input_0, input_1;
     float input_2;
     printf("Enter three positive numbers: ");
     scanf("%d %x %f", &input_0, &input_1, &input_2);
     printf("%o, %o, %o\n", input_0, input_1, (int)
input_2);
     printf("%#x, %#x, %#x\n", input_0, input_1, (int)
input_2);
     printf("%.2f, %.2f, %.2f\n", (float) input_0, (float)
input_1, input_2);
     return 0;
}
```

Requirement:

For following tasks, you also need to show any helper function(s) your function/program uses.

Task 6: **Basic Sorting:** Write a **function** that uses qsort() to sort an array of integers in ascending order. The function should take an array and its size as arguments. Hint:

```
int compare(const void* a, const void* b) {
     return *(int*) a - *(int*) b;
}

void sort(int array[], int size) {
     qsort(array, size, sizeof(int), compare);
}
```

Task 7: **Descending Order Sorting:** Modify the previous solution to sort the integers in descending order using qsort().

```
int compare(const void* a, const void* b) {
     return *(int*) a - *(int*) b;
}

void sort(int array[], int size) {
     qsort(array, size, sizeof(int), compare);
}
```

Task 8: **Sorting Structures:** Given a structure Person with name (string: char[]) and age (integer), write a **program** that sorts an array of Person structures by age in ascending order first, and then by name in descending order, using qsort().

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

typedef struct {
     int  age;
     char name[128];
} Person;

int compare_person(const void* a, const void* b) {
     Person* person_a = (Person*) a;
     Person* person_b = (Person*) b;
     if (person_a->age == person_b->age) {
     return strcmp(person_b->name, person_a->name);
     } else {
     return person_a->age - person_b->age;
     }
}


void print_array(Person persons[], size_t size) {
     for (size_t i = 0; i < size; ++i) {
     printf("%s: %d\n", persons[i].name, persons[i].age);
     }
}

// NOTE: comparator functions take in two const void
pointers
//       then we have to cast them into the correct
//       types, then get the value.
int main(void) {
     Person person_0, person_1, person_2, person_3;
     person_0.age = 2; sscanf("aaa", "%s", person_0.name);
     person_1.age = 0; sscanf("bbb", "%s", person_1.name);
     person_2.age = 1; sscanf("ccc", "%s", person_2.name);
     person_3.age = 1; sscanf("ddd", "%s", person_3.name);
```

```
    Person persons[] = {person_0, person_1, person_2,
person_3};
    size_t num_persons = 4;

    qsort(persons, num_persons,
        sizeof(Person), compare_person);

    print_array(persons, num_persons);
    return 0;
}
```

Task 9: **Custom Comparator:** Write a custom comparator **function** for qsort() that sorts an array of strings in lexicographical order, but with the strings sorted by length in descending order first, and then alphabetically (in ascending order).

```
int compare_custom(const void* a, const void* b) {
    const char* _a = *(const char**) a;
    const char* _b = *(const char**) b;
    size_t _a_len = strlen(_a);
    size_t _b_len = strlen(_b);

    if (_a_len > _b_len) {
        return -1;
    } else if (_a_len < _b_len) {
        return 1;
    } else {
        return strcmp(_a, _b);
    }
}
```

Task 10: **Sorting a Subarray:** Write a **program** that sorts only a part of an array using qsort(). The function should take an array, its start index, end index, and size as parameters and sort the elements within the specified range.

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

int compare(const void* a, const void* b) {
     return *(int*) a - *(int*) b;
}

// inclusive
void sort_subarray(int ints[], size_t start,
     size_t end, size_t size) {
     if (end >= size) {
          printf("WARNING: end index too large. \n");
          end = size - 1;
     }

     if (start >= size) {
          printf("WARNING: start index is too small or too
large. \n");
          start = 0;
     }

     // swap
     if (start > end) {
          size_t hold = start;
          start = end;
          end = hold;
     }


     printf("%lu, %lu\n", start, end);
     qsort(ints + start, end - start + 1,
          sizeof(int), compare);
}

int main(void) {
     int ints[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0};
```

```
    size_t size = sizeof(ints)/sizeof(int);
    sort_subarray(ints, 3, 12, size);
    for (size_t i = 0; i < size; ++i) {
    printf("%i\n", ints[i]);
    }

    return 0;
}
```