Submission details:

- Swoyam Pokharel
- Workshop 01

Modify the following program to use a continue statement to skip printing the number 5 and a break statement to stop the loop once the number reaches 8

```c
#include <stdio.h>

void main() {
    for(int n = 0; n < 10; n++){
        if (n == 5) continue; // Added this
        if (n == 8) break; // Added this
        printf("%d\n", n);
    }

}
```

stdout

```
[wizard@archlinux Week1]$ gcc main.c && ./a.out
0
1
2
3
4
6
7
[wizard@archlinux Week1]$
```

Write a C program that performs the following:

- Asks the user to input 5 integers.
- Passes the array to a function that finds and returns the maximum value.
- Passes the array to a function that sorts the array in ascending order using

a simple bubble sort algorithm.

- Create a user-defined header file array_operation.h that declares the functions for finding the maximum value and sorting the array.
- Create a separate implementation file array_operation.c to define the functions declared in the header file.

main.c

```c
#include <stdio.h>
#include "./array_ops.h"

void main() {
    int arr[5];
    printf("Enter 5 integers: \n");
    for(int i = 0; i < 5; i++) {
        scanf("%d", &arr[i]);
    }

    int max = find_max(arr, 5);
    printf("max: %d\n", max);
    bubble_sort(arr, 5);
    printf("sorted:");
    for(int i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
    }
}
```

array_ops.h

```c
#define ARRAY_OPERATION_H

int find_max(int *arr, int size);
void bubble_sort(int *arr, int size);
```

array_ops.c

```c
#include "./array_ops.h"

int find_max(int *a, int n) {
    int m = a[0];
    for(int i = 1; i < n; i++) if (a[i] > m) m = a[i];
    return m;
}

void bubble_sort(int *arr, int size) {
    for(int i = 0; i < size-1; i++) {
        for(int j = 0; j < size-i-1; j++){
```

```
            if(arr[j] > arr[j+1]) {
                int tmp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = tmp;
            }
        }
    }
}
```

stdout

```
[wizard@archlinux Week1]$ gcc main.c array_ops.c
[wizard@archlinux Week1]$ ./a.out
Enter 5 integers:
5
3
9
1
2
max: 9
sorted:1 2 3 5 9
[wizard@archlinux Week1]$
```

Define a structure to store information about a student (e.g., name, age, and GPA). Write a program that:

- Accepts input from the user to fill in the details for 3 students.
- Displays the details for each student after all inputs have been provided.

```c
int main() {
    struct Student students[3];

    for (int i = 0; i < 3; i++) {
        printf("Enter details for student %d:\n", i+1);

        printf("Name: ");
        scanf(" %[^\n]", students[i].name);

        printf("Age: ");
        scanf("%d", &students[i].age);

        printf("GPA: ");
        scanf("%f", &students[i].gpa);
        printf("\n");
    }

    printf("Student Details:\n");
    for (int i = 0; i < 3; i++) {
```

```
        printf("Student %d:\n", i+1);
        printf("Name: %s\n", students[i].name);
        printf("Age: %d\n", students[i].age);
        printf("GPA: %.2f\n\n", students[i].gpa);
    }

    return 0;
}
```

stdout

```
[wizard@archlinux Week1]$ gcc main.c && ./a.out
Enter details for student 1:
Name: Jhon
Age: 29
GPA: 3.9

Enter details for student 2:
Name: Bob
Age: 11
GPA: 3.8

Enter details for student 3:
Name: Charlie
Age: 32
GPA: 4.0

Student Details:
Student 1:
Name: Jhon
Age: 29
GPA: 3.90

Student 2:
Name: Bob
Age: 11
GPA: 3.80

Student 3:
Name: Charlie
Age: 32
GPA: 4.00

[wizard@archlinux Week1]$
```

The following code prints out the value of an int variable and a string (char *):

```c
#include <stdio.h>
void main(int argc, char *argv[]) {
    int age = 10; char *name = "Hiran";
    printf("Hello %s, you are %d years old.", name, age);
}
```

Now modify the program so that it uses the command line arguments to supply name and age. i.e. it uses the argc and argv arguments/ parameters.

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

int main(int argc, char *argv[]) {
    if (argc < 3) {
        printf("Usage: %s <name> <age>\n", argv[0]);
        return 1;
    }

    char *name = argv[1];
    int age = atoi(argv[2]);

    printf("%s, you are %d years old.\n", name, age);

    return 0;
}
```

stdout

```c
[wizard@archlinux Week1]$ gcc main.c && ./a.out Wizard 16
Wizard, you are 16 years old.
[wizard@archlinux Week1]$
```

Now modify the program again so that it uses the scanf() function to get input from the user for the name and age.

```c
#include <stdio.h>
int main() {
    char name[50];  // buffer to store the name
    int age;

    // Ask the user for input
    printf("Enter your name: ");
    scanf(" %[^\n]", name);
```

```
    printf("Enter your age: ");
    scanf("%d", &age);

    printf("Hello %s, you are %d years old.\n", name, age);

    return 0;
}
```

stdout

```
[wizard@archlinux Week1]$ ./a.out
Enter your name: Wizard
Enter your age: 16
Hello Wizard, you are 16 years old.
[wizard@archlinux Week1]$
```

The following code count the integer variable n from 0 to 9 and prints out “Odd” if n is even and just the value of n if it is even: #include <stdio.h>

```c
void main(int argc, char *argv[])
{
    for(int n =0; n <10; n++){
        if(n % 2 == 1){
            printf("%d is Odd\n", n);
        }
        else{
            printf("%d\n", n);
        }
    }
}
```

When you run the program, it should output the following:

```
0
1 is Odd
2
3 is Odd
4
5 is Odd
6
7 is Odd
8
9 is Odd
```

Now modify the program so that it counts the variable n from 1 to 100 and, if n is a multiple of 2 (eg. 2, 4, 6, etc), it would print out the word “Bish”, and if n is a multiple of 3 (eg. 3, 6, 9. 12 etc), it would print out the word “Bash”, and if n is a multiple of 5 (eg. 5, 10, 15 etc), it would print out the word “Bosh”.

However, if n is a multiple of 2 and 3 (eg. 6), it would print out the words “BishBash”, and if n is a multiple of 2 and 5 (eg. 10), it would print out the words “BishBosh”, and if n is a multiple of 3 and 5 (eg. 15), it would print out the words “BashBosh”. Finally, if n is a multiple of 2, 3 and 5 (eg. 30), it would print out the words “BishBashBosh”.

```c
int main() {
    for(int n=1;n<=100;n++){
        if(n%2==0) printf("Bish");
        if(n%3==0) printf("Bash");
        if(n%5==0) printf("Bosh");
        if(n%2 && n%3 && n%5) printf("%d", n);
        printf("\n");
    }
}
```

stdout

```
[wizard@archlinux Week1]$ gcc main.c && ./a.out
1
Bish
Bash
Bish
Bosh
BishBash
7
Bish
Bash
BishBosh
11
BishBash
13
Bish
BashBosh
Bish
17
BishBash
19
BishBosh
Bash
Bish
23
BishBash
```

```
Bosh
Bish
Bash
Bish
29
BishBashBosh
31
Bish
Bash
Bish
Bosh
BishBash
37
Bish
Bash
BishBosh
41
BishBash
43
Bish
BashBosh
Bish
47
BishBash
49
BishBosh
Bash
Bish
53
BishBash
Bosh
Bish
Bash
Bish
59
BishBashBosh
61
Bish
Bash
Bish
Bosh
BishBash
67
Bish
Bash
BishBosh
71
BishBash
73
Bish
BashBosh
Bish
77
BishBash
```

```
79
BishBosh
Bash
Bish
83
BishBash
Bosh
Bish
Bash
Bish
89
BishBashBosh
91
Bish
Bash
Bish
Bosh
BishBash
97
Bish
Bash
BishBosh
[wizard@archlinux Week1]$
```

WAP to swap the values of 2 variables using a function.

```c
#include <stdio.h>

void swap(int *a, int *b) {
    int t= *a;
    *a = *b;
    *b = t;
}

int main() {
    int a = 3, b = 0;
    printf("Initial Values, a: %d, b: %d\n", a, b);
    swap(&a,&b);
    printf("After Swap, a: %d, b: %d\n", a, b);
}
```

stdout

```
[wizard@archlinux Week1]$ gcc main.c && ./a.out
Initial Values, a: 3, b: 0
After Swap, a: 0, b: 3
[wizard@archlinux Week1]$
```

The following program fills an int array of size 10 and fills it with random numbers and prints them out:

```c
#include <stdio.h>
#include <stdlib.h>
void main(int argc, char *argv[]) {
    int numbers[10];
    for (int i=0; i < 10; i++){
        numbers[i] = rand();
        printf("%d is %d\n", i, numbers[i]);
    }
}
```

Now modify it to will ask the user for a number between 1 and 50, and then use the C function malloc() to allocate an `int` array of that size, fill it with random numbers and print out the value of each element of that array.

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

int main() {
    int n;

    printf("Enter a number between 1 and 50: ");
    scanf("%d", &n);

    int *numbers = malloc(n * sizeof(int));
    for(int i = 0; i < n; i++) {
        numbers[i] = rand();
        printf("%d is %d\n", i, numbers[i]);
    }
    free(numbers);
    return 0;
}
```

stdout

```
[wizard@archlinux Week1]$ gcc main.c && ./a.out
Enter a number between 1 and 50: 9
0 is 323198923
1 is 627256817
2 is 955659829
3 is 831220215
```

```
4 is 1810005391
5 is 528160378
6 is 288462182
7 is 682833526
8 is 634251420
[wizard@archlinux Week1]$
```

The following code creates 2 threads in a program and counts to 10 in each thread :

```c
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
void *threadA(void *p){
    for(int i=0; i<10; i++){
        printf("Thread ID %ld: i=%d\n", pthread_self(), i);
        usleep(1000);
    }
}

void *threadB(void *p){
    for(int i=0; i<10; i++){
        printf("Thread ID %ld: i=%d\n", pthread_self(), i);
        usleep(1000);
    }
}

void main(){
    pthread_t thrID1, thrID2;
    pthread_create(&thrID1, NULL, threadA, NULL);
    pthread_create(&thrID2, NULL, threadB, NULL);
    pthread_join(thrID1, NULL);
    pthread_join(thrID2, NULL);
}
```

Modify the program to accept a command line argument to specific the number of threads, and then create that many threads dynamically to run

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

void *threadFunc(void *arg) {
    int id = *(int *)arg;
    free(arg);
```

```c
    for(int i = 0; i < 10; i++) {
        printf("Thread %d (ID %ld): i=%d\n", id, pthread_self(), i);
        usleep(1000);
    }
}

int main(int argc, char *argv[]) {
    int numThreads = atoi(argv[1]);
    pthread_t *threads = malloc(numThreads * sizeof(pthread_t));

    for(int i = 0; i < numThreads; i++) {
        int *id = malloc(sizeof(int));
        *id = i;
        pthread_create(&threads[i], NULL, threadFunc, id);
    }

    for(int i = 0; i < numThreads; i++) {
        pthread_join(threads[i], NULL);
    }

    free(threads);
    return 0;
}
```

stdout

```
[wizard@archlinux Week1]$ gcc main.c && ./a.out 4
Thread 0 (ID 140361077511872): i=0
Thread 2 (ID 140361060726464): i=0
Thread 1 (ID 140361069119168): i=0
Thread 3 (ID 140361052333760): i=0
Thread 0 (ID 140361077511872): i=1
Thread 2 (ID 140361060726464): i=1
Thread 1 (ID 140361069119168): i=1
Thread 3 (ID 140361052333760): i=1
Thread 0 (ID 140361077511872): i=2
Thread 2 (ID 140361060726464): i=2
Thread 1 (ID 140361069119168): i=2
Thread 3 (ID 140361052333760): i=2
Thread 0 (ID 140361077511872): i=3
Thread 1 (ID 140361069119168): i=3
Thread 2 (ID 140361060726464): i=3
Thread 3 (ID 140361052333760): i=3
Thread 0 (ID 140361077511872): i=4
Thread 1 (ID 140361069119168): i=4
Thread 2 (ID 140361060726464): i=4
Thread 3 (ID 140361052333760): i=4
Thread 0 (ID 140361077511872): i=5
Thread 2 (ID 140361060726464): i=5
Thread 1 (ID 140361069119168): i=5
Thread 3 (ID 140361052333760): i=5
Thread 0 (ID 140361077511872): i=6
Thread 2 (ID 140361060726464): i=6
```

```
Thread 1 (ID 140361069119168): i=6
Thread 3 (ID 140361052333760): i=6
Thread 0 (ID 140361077511872): i=7
Thread 2 (ID 140361060726464): i=7
Thread 1 (ID 140361069119168): i=7
Thread 3 (ID 140361052333760): i=7
Thread 0 (ID 140361077511872): i=8
Thread 2 (ID 140361060726464): i=8
Thread 1 (ID 140361069119168): i=8
Thread 3 (ID 140361052333760): i=8
Thread 0 (ID 140361077511872): i=9
Thread 2 (ID 140361060726464): i=9
Thread 1 (ID 140361069119168): i=9
Thread 3 (ID 140361052333760): i=9
[wizard@archlinux Week1]$
```