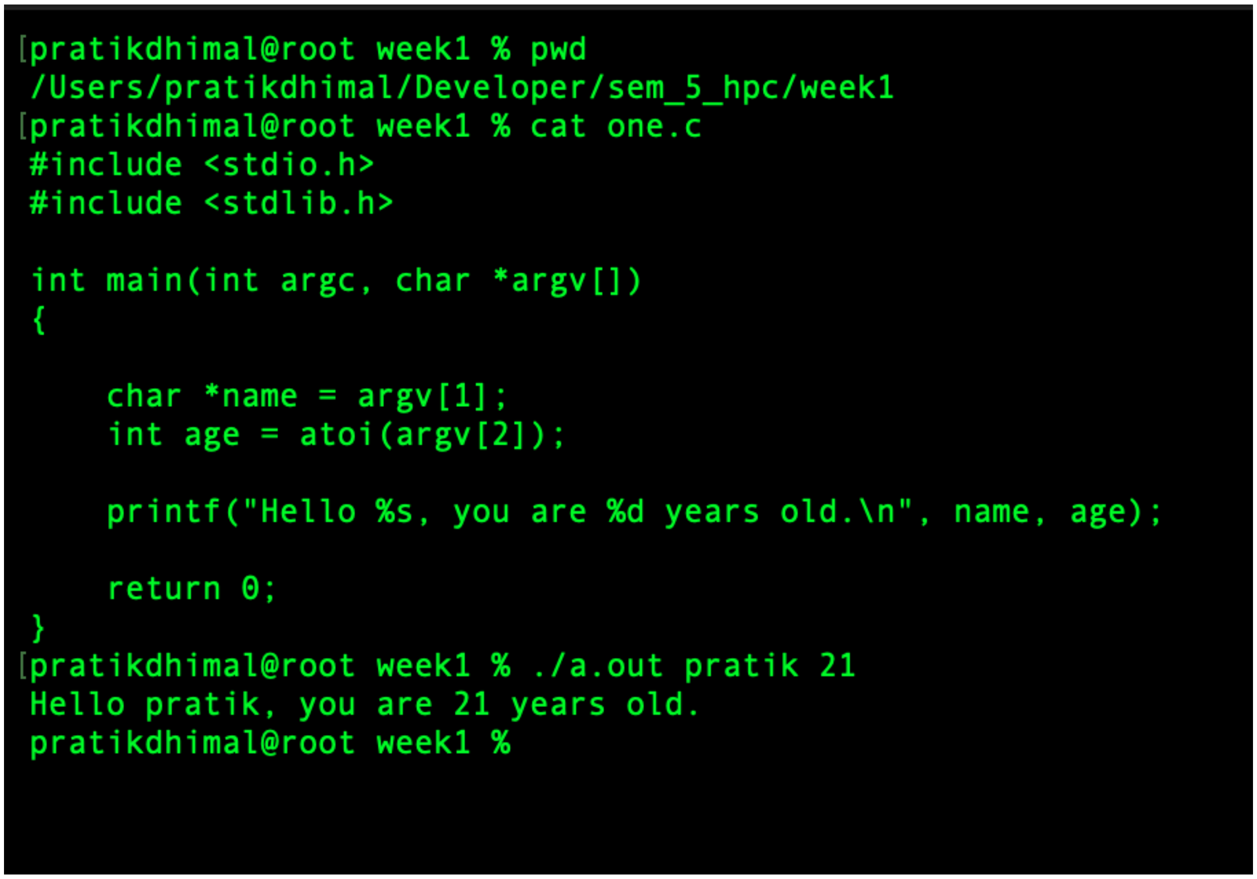
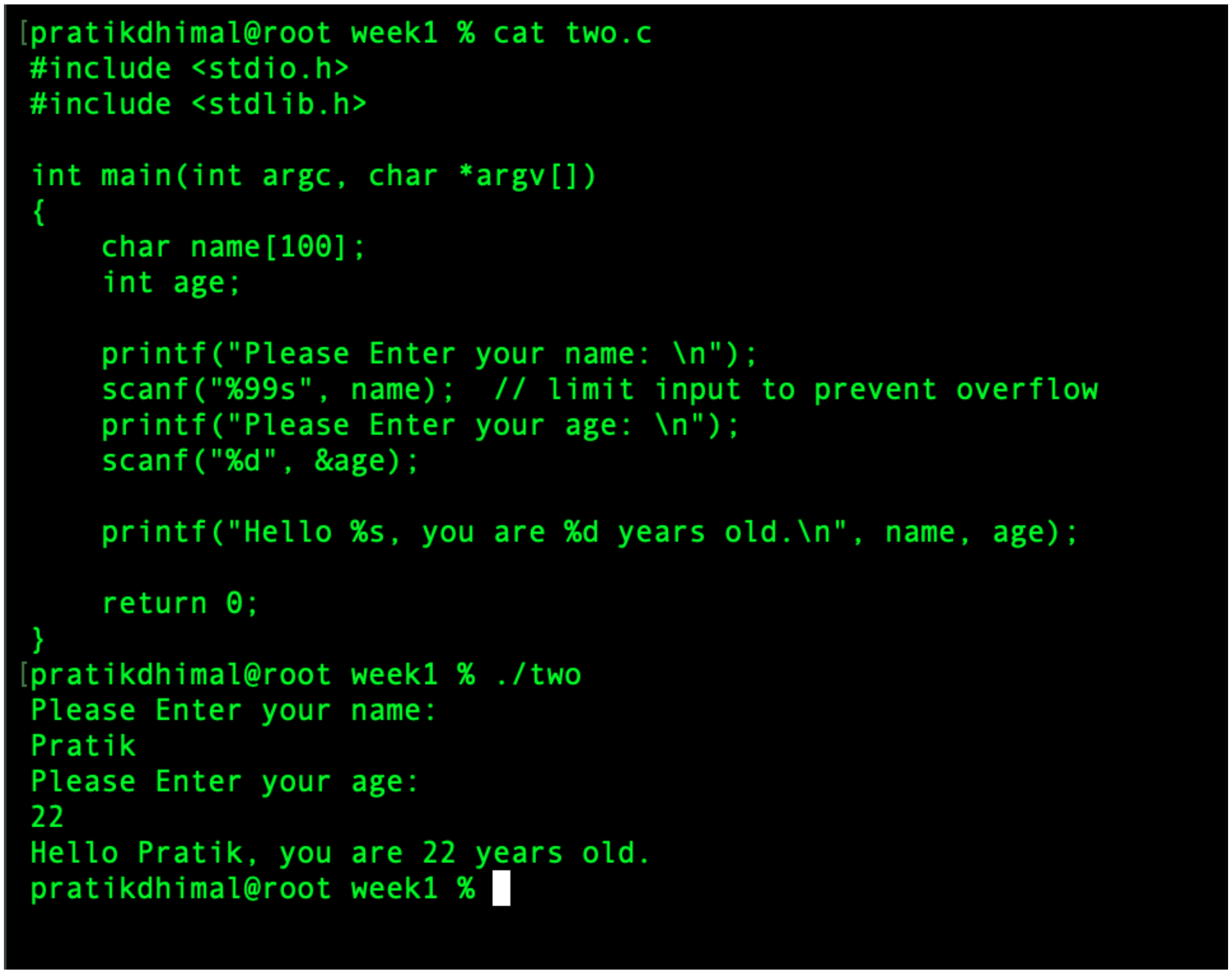


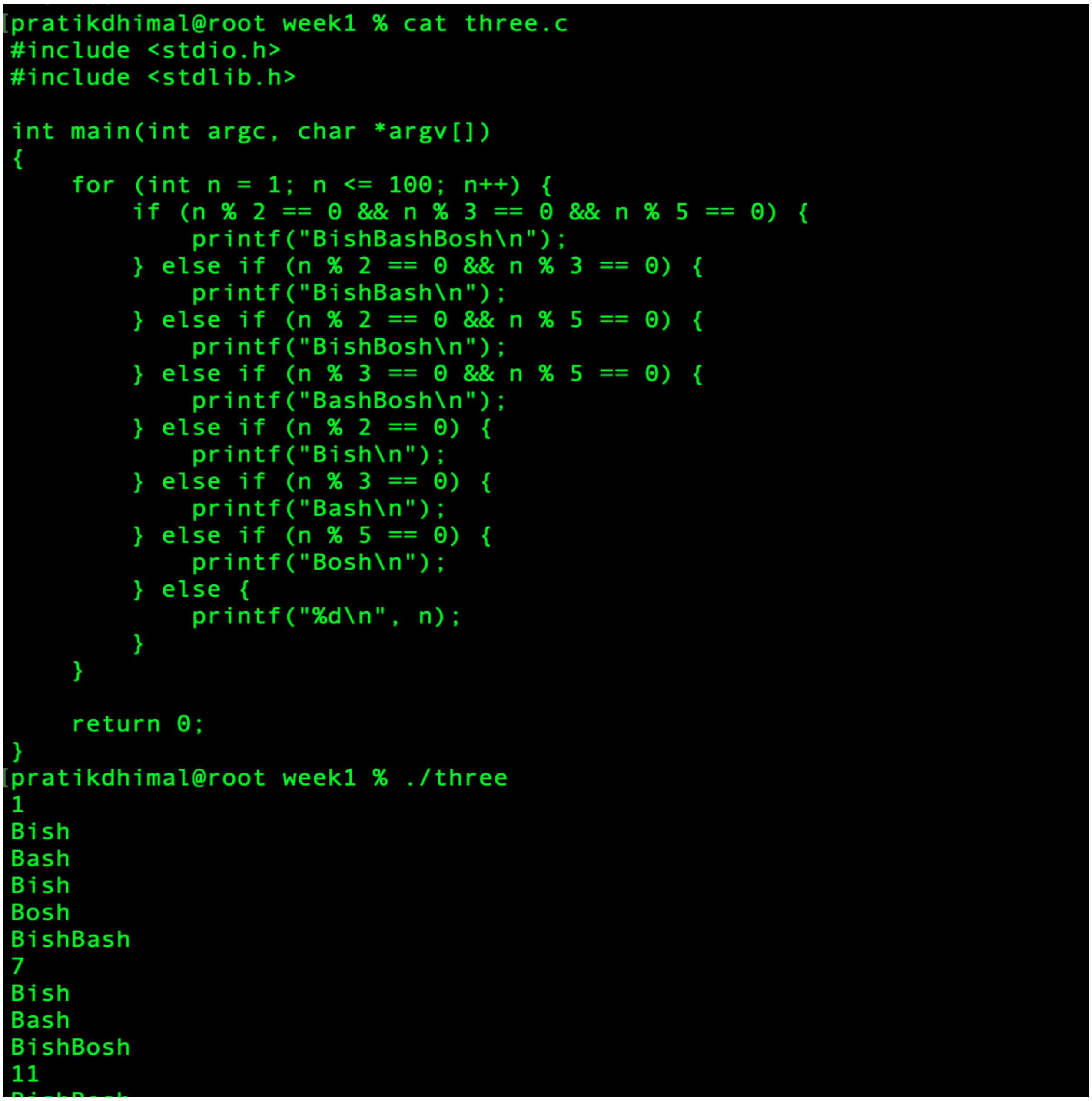
```
[pratikdhimal@root week1 % pwd
/Users/pratikdhimal/Developer/sem_5_hpc/week1
[pratikdhimal@root week1 % cat one.c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{

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

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

    return 0;
}
[pratikdhimal@root week1 % ./a.out pratik 21
Hello pratik, you are 21 years old.
pratikdhimal@root week1 %
```

⇒ Here, I have used **argv** to get the values from the CLI. All the values from CLI are in string format so I used **atoi()** to convert string to int.



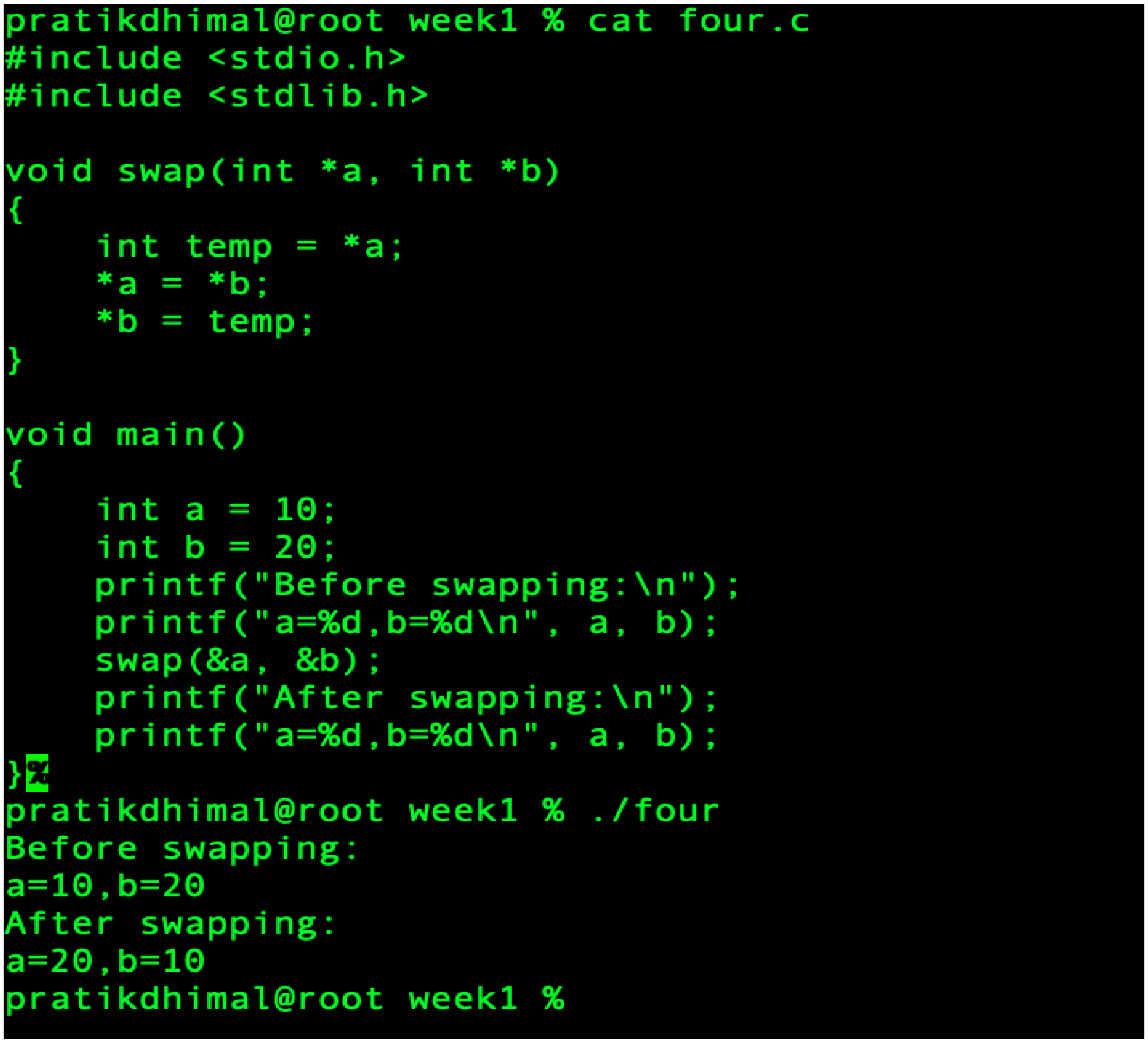
```
[pratikdhimal@root week1 % cat two.c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    char name[100];
    int age;

    printf("Please Enter your name: \n");
    scanf("%99s", name);  // limit input to prevent overflow
    printf("Please Enter your age: \n");
    scanf("%d", &age);

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

    return 0;
}
[pratikdhimal@root week1 % ./two
Please Enter your name:
Pratik
Please Enter your age:
22
Hello Pratik, you are 22 years old.
pratikdhimal@root week1 % 
```

⇒ Here I have declared variables for name and age and then used **scanf()** to get input from the user. Lastly, I have just displayed the output using **printf()**



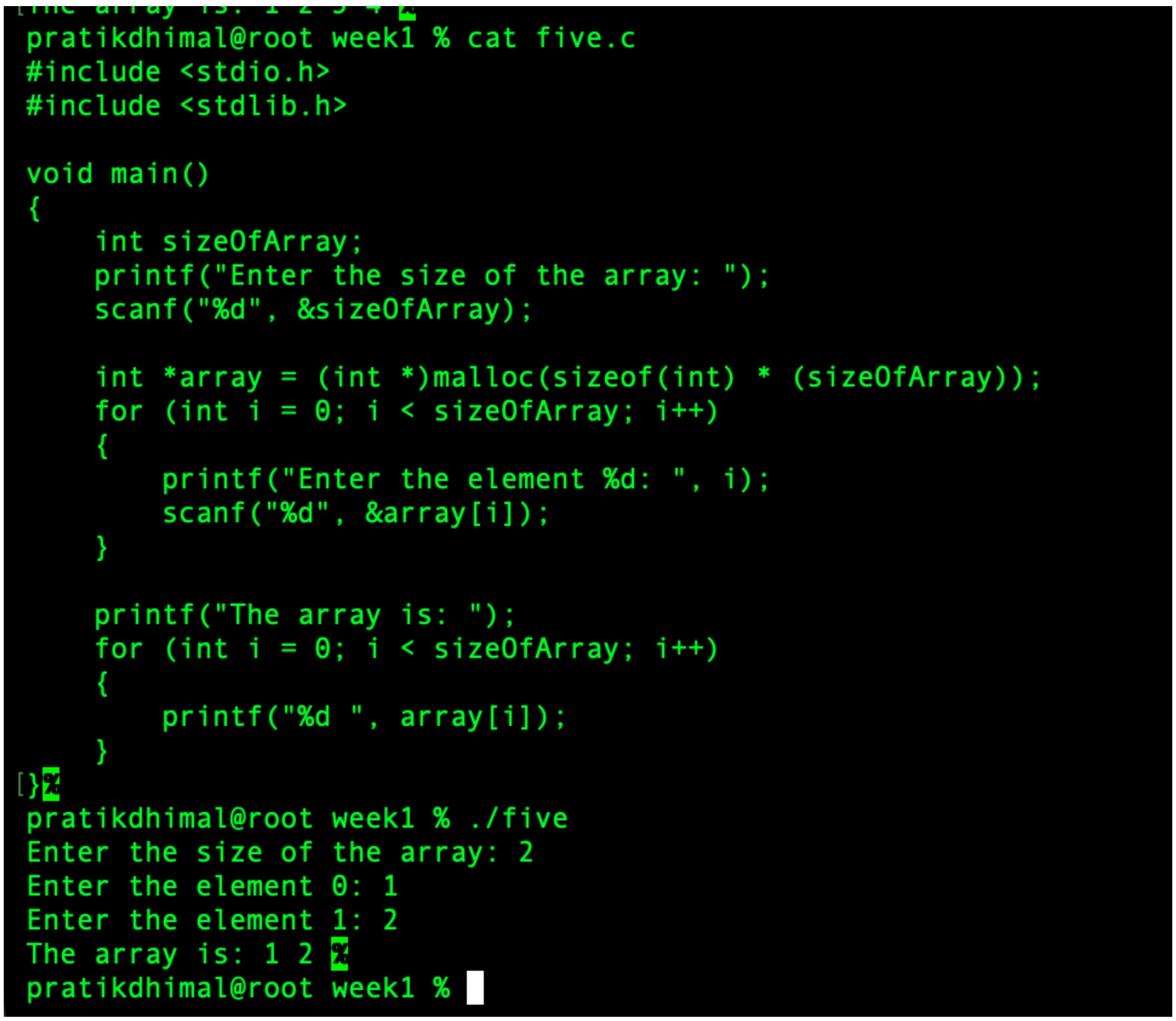
```
[pratikdhimal@root week1 % cat three.c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    for (int n = 1; n <= 100; n++) {
        if (n % 2 == 0 && n % 3 == 0 && n % 5 == 0) {
            printf("BishBashBosh\n");
        } else if (n % 2 == 0 && n % 3 == 0) {
            printf("BishBash\n");
        } else if (n % 2 == 0 && n % 5 == 0) {
            printf("BishBosh\n");
        } else if (n % 3 == 0 && n % 5 == 0) {
            printf("BashBosh\n");
        } else if (n % 2 == 0) {
            printf("Bish\n");
        } else if (n % 3 == 0) {
            printf("Bash\n");
        } else if (n % 5 == 0) {
            printf("Bosh\n");
        } else {
            printf("%d\n", n);
        }
    }

    return 0;
}
[pratikdhimal@root week1 % ./three
1
Bish
Bash
Bish
Bosh
BishBash
7
Bish
Bash
BishBosh
11
```

⇒ Here I used a **loop** to loop from 1 to 100 and added **if, else if, and else** conditions to check the required condition and displayed the output.



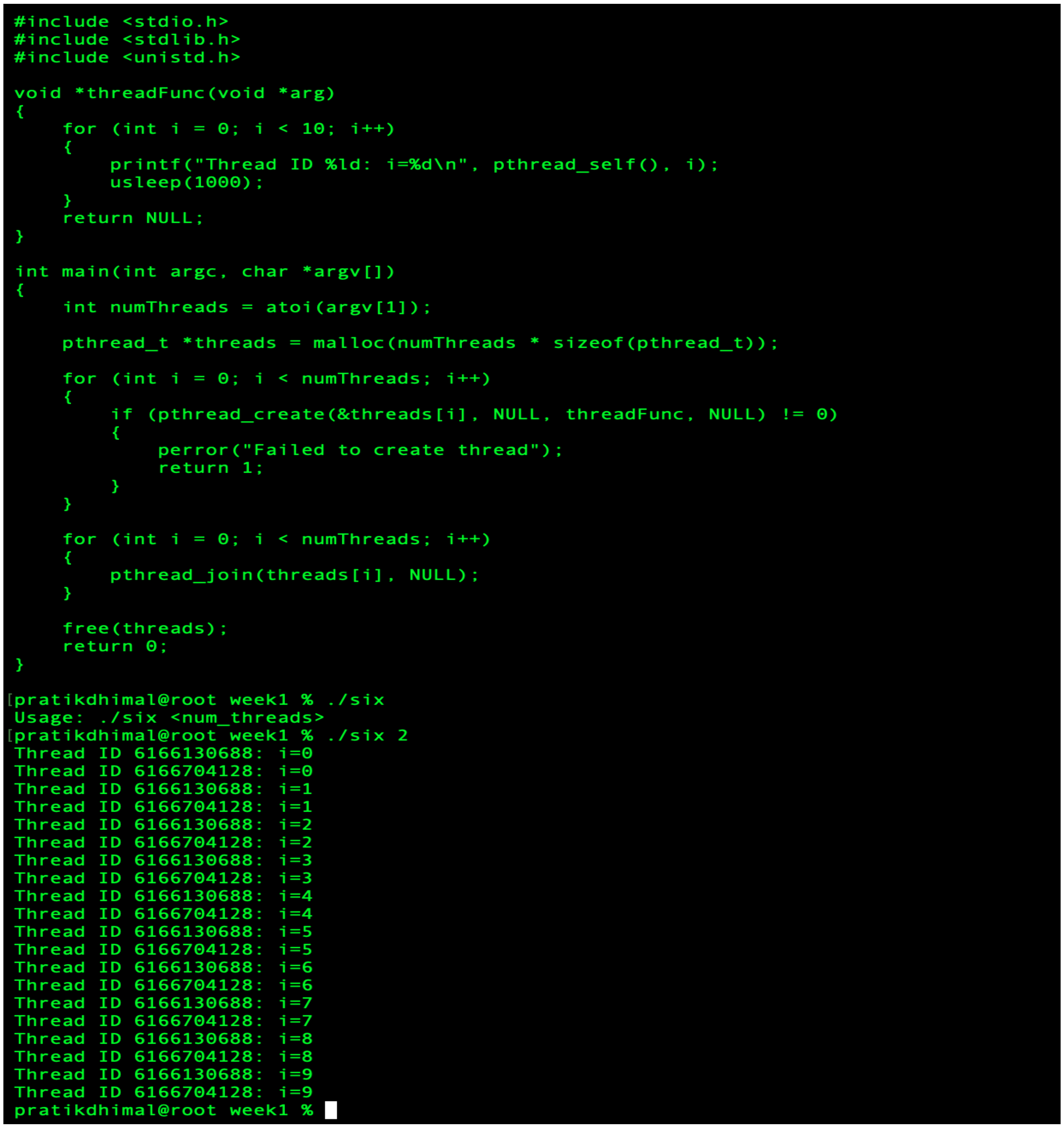
```
pratikdhimal@root week1 % cat four.c
#include <stdio.h>
#include <stdlib.h>

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

void main()
{
    int a = 10;
    int b = 20;
    printf("Before swapping:\n");
    printf("a=%d,b=%d\n", a, b);
    swap(&a, &b);
    printf("After swapping:\n");
    printf("a=%d,b=%d\n", a, b);
}%
pratikdhimal@root week1 % ./four
Before swapping:
a=10,b=20
After swapping:
a=20,b=10
pratikdhimal@root week1 %
```

⇒Here, I have made a function **swap()** which takes the memory **address of two integers.** This function then introduces a temp variable to store the value of one integer and then swap the other with the first one. Lastly, the value form temporary variable is set to the swapped one.Hence, the integers are swapped.



```
[The array is: 1 2 3 4 %
pratikdhimal@root week1 % cat five.c
#include <stdio.h>
#include <stdlib.h>

void main()
{
    int sizeOfArray;
    printf("Enter the size of the array: ");
    scanf("%d", &sizeOfArray);

    int *array = (int *)malloc(sizeof(int) * (sizeOfArray));
    for (int i = 0; i < sizeOfArray; i++)
    {
        printf("Enter the element %d: ", i);
        scanf("%d", &array[i]);
    }

    printf("The array is: ");
    for (int i = 0; i < sizeOfArray; i++)
    {
        printf("%d ", array[i]);
    }
[}%
pratikdhimal@root week1 % ./five
Enter the size of the array: 2
Enter the element 0: 1
Enter the element 1: 2
The array is: 1 2 %
pratikdhimal@root week1 %
```

⇒ Here, I have asked input form users and created memory based on that using **malloc()** and then a loop is used to add values to those allocated memory.



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

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

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++)
    {
        if (pthread_create(&threads[i], NULL, threadFunc, NULL) != 0)
        {
            perror("Failed to create thread");
            return 1;
        }
    }

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

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

```
[pratikdhimal@root week1 % ./six
 Usage: ./six <num_threads>
[pratikdhimal@root week1 % ./six 2
 Thread ID 6166130688: i=0
 Thread ID 6166704128: i=0
 Thread ID 6166130688: i=1
 Thread ID 6166704128: i=1
 Thread ID 6166130688: i=2
 Thread ID 6166704128: i=2
 Thread ID 6166130688: i=3
 Thread ID 6166704128: i=3
 Thread ID 6166130688: i=4
 Thread ID 6166704128: i=4
 Thread ID 6166130688: i=5
 Thread ID 6166704128: i=5
 Thread ID 6166130688: i=6
 Thread ID 6166704128: i=6
 Thread ID 6166130688: i=7
 Thread ID 6166704128: i=7
 Thread ID 6166130688: i=8
 Thread ID 6166704128: i=8
 Thread ID 6166130688: i=9
 Thread ID 6166704128: i=9
 pratikdhimal@root week1 %
```

⇒ Here input form user is taken to create the number of threads. Then based on that loop is runned to create threads function as per the users response. Then another loop is used to create and joint the thread