**University of Wolverhampton**

**School of Mathematics and Computer Science**

**Student Number: 2408869**

**Name: Narayan Lohani**

**6CS005 High Performance Computing Week 1 Workshop**

**Revision on C and Multithreading  
  
Tasks – Basic C Syntax**

**Table of Figures**

[Figure 1: Source Code of program 1 2](#_Toc210810532)

[Figure 2: Output of Program 1 2](#_Toc210810533)

[Figure 3: Output of Program 2 4](#_Toc210810534)

[Figure 4: Source Code Program 2 4](#_Toc210810535)

[Figure 5: Array Operation Header file Program 2 4](#_Toc210810536)

[Figure 6: Array Operation Source Code Program 2 4](#_Toc210810537)

[Figure 7: Source Code of Program 3 6](#_Toc210810538)

[Figure 8: Output of Program 3 7](#_Toc210810539)

[Figure 9: Source Code Program 4 8](#_Toc210810540)

[Figure 10: Output of Program 4 8](#_Toc210810541)

[Figure 11: Source Code of Program 5 9](#_Toc210810542)

[Figure 12: Output of Program 5 9](#_Toc210810543)

[Figure 13: Output of program 6 11](#_Toc210810544)

[Figure 14: Source Code of Program 6 11](#_Toc210810545)

[Figure 15: Source Code of Program 7 12](#_Toc210810546)

[Figure 16: Output of program 7 12](#_Toc210810547)

[Figure 17: Output of program 8 13](#_Toc210810548)

[Figure 18: Source Code of program 8 13](#_Toc210810549)

[Figure 19: Source code of program 9 15](#_Toc210810550)

[Figure 20: Output of program 9 16](#_Toc210810551)

1. 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.

#include <stdio.h>

void main() {

for(int n = 0; n < 10; n++){

printf("%d\n", n);

}

}

Figure 1: Source Code of program 1

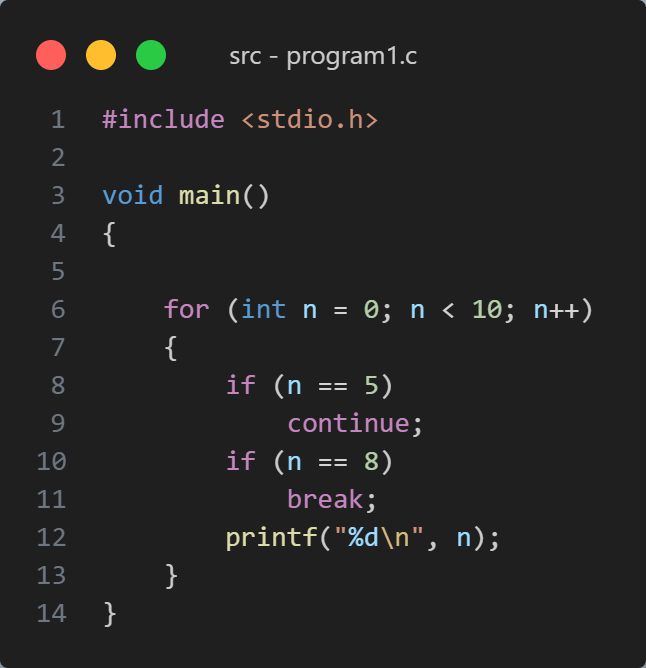
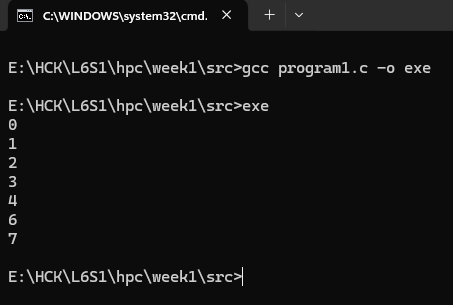


Figure 2: Output of Program 1



1. 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.

#include <stdio.h>

int find\_max(int arr[], int size);

void main() {

int arr[5];

printf("Enter 5 integers: \n");

for (int i = 0; i < 5; i++) {

scanf("%d", &arr[i]);

}

int max\_val = find\_max(arr, 5);

printf("The maximum value is: %d\n", max\_val);

}

int find\_max(int arr[], int size) {

// Function logic here

}

void sort\_ascending(int arr[], int size) {

// Function logic here

}

Figure 3: Output of Program 2

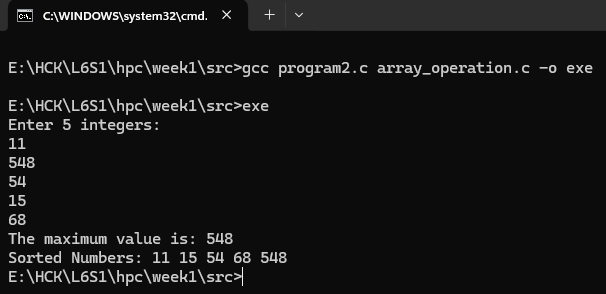


Figure 4: Source Code Program 2

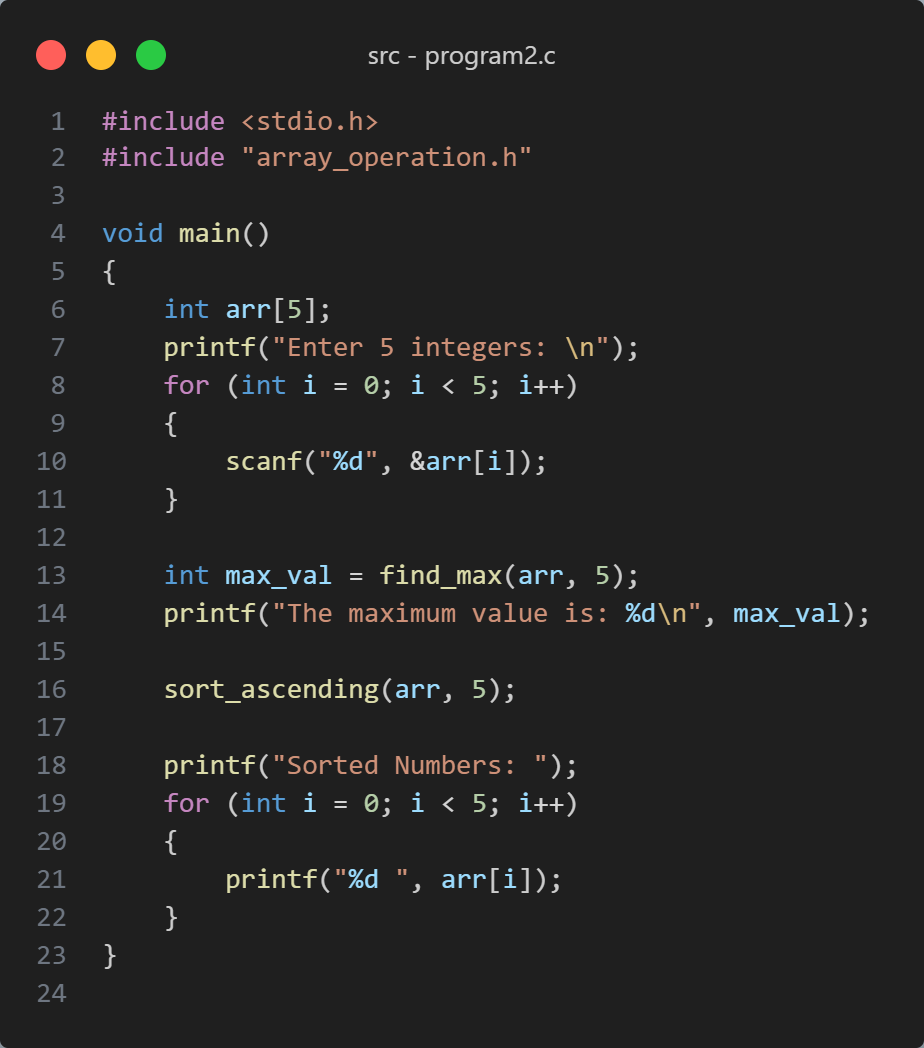


Figure 5: Array Operation Header file Program 2

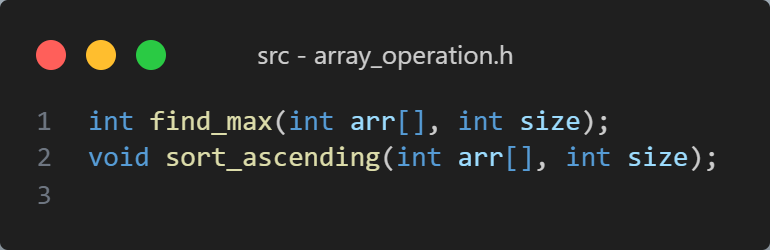
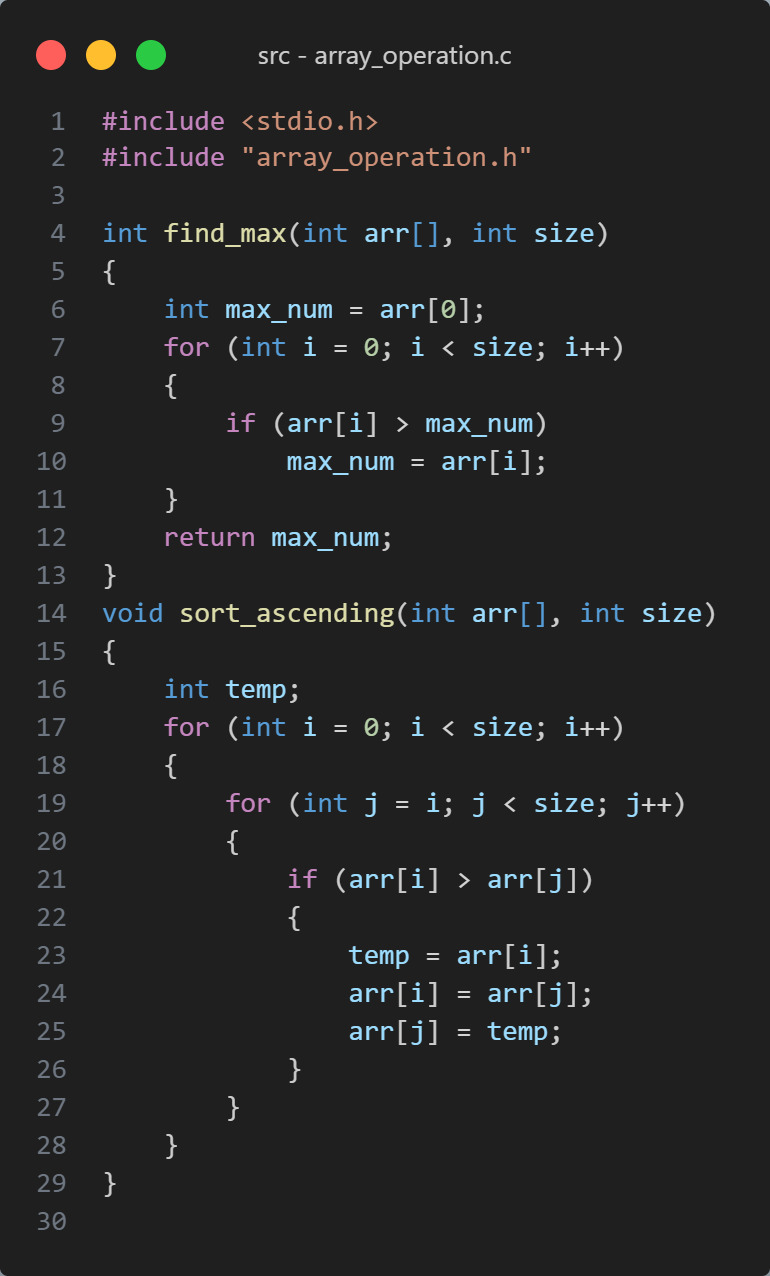


Figure 6: Array Operation Source Code Program 2



1. Define a structure to store information about a student (e.g., name, age, and GPA). Write a program that:
2. Accepts input from the user to fill in the details for 3 students.
3. Displays the details for each student after all inputs have been provided.

*Expected Output:*Enter the details of student 1

Name: John

Age: 20

GPA: 3.6

Enter the details of student 2

Name: Alice

Age: 21

GPA: 3.8

Enter the details of student 3

Name: Bob

Age: 19

GPA: 3.7

Student Details:

Name: John, Age: 20, GPA: 3.60

Name: Alice, Age: 21, GPA: 3.80

Name: Bob, Age: 19, GPA: 3.70

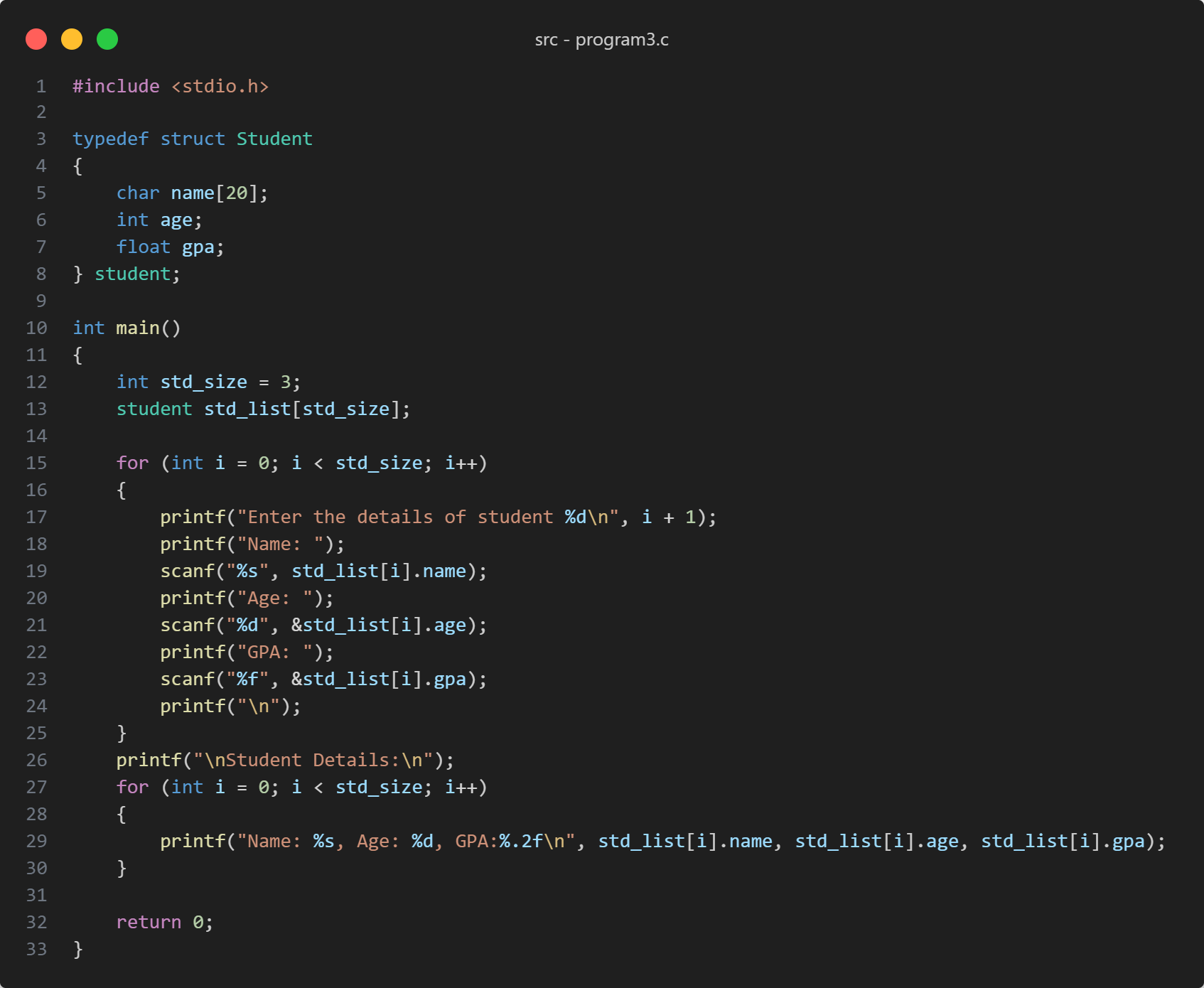


Figure 7: Source Code of Program 3

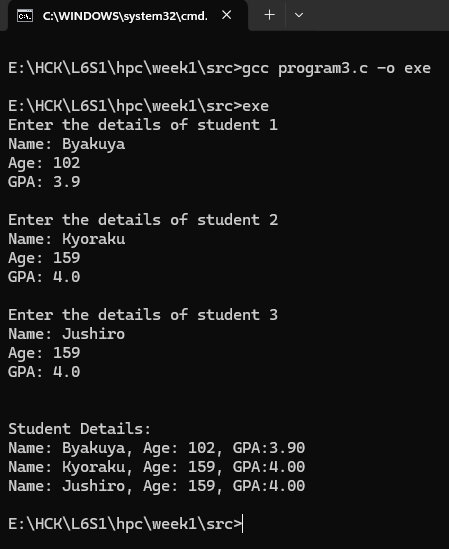


Figure 8: Output of Program 3

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

4. 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.

When you run it, it should produce the following:

**./myprog Jeffrey 100**

**Hello Jeffrey, you are 100 years old.**

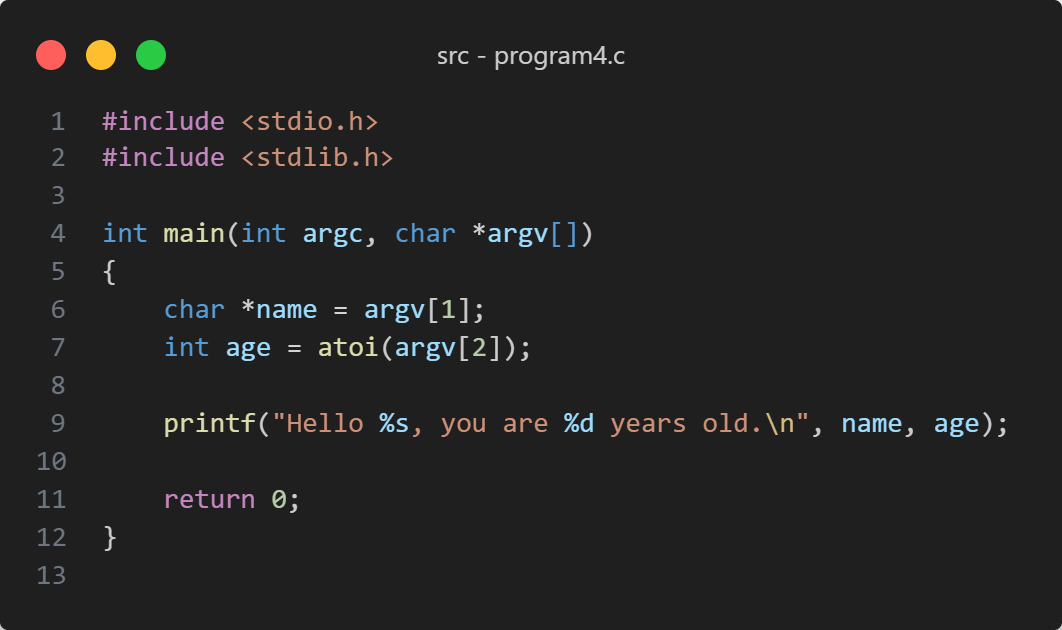


Figure 9: Source Code Program 4

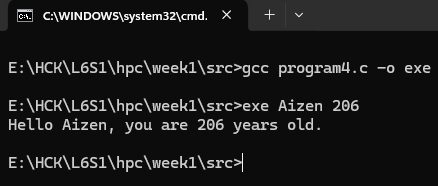


Figure 10: Output of Program 4

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



Figure 11: Source Code of Program 5

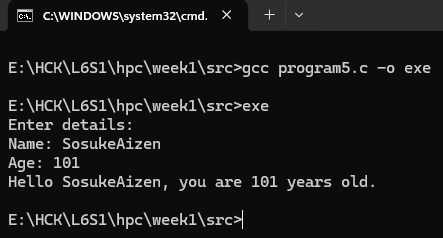


Figure 12: Output of Program 5

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

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

6. 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”.

When you run the program, it will produce something like this:

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

Figure 13: Output of program 6

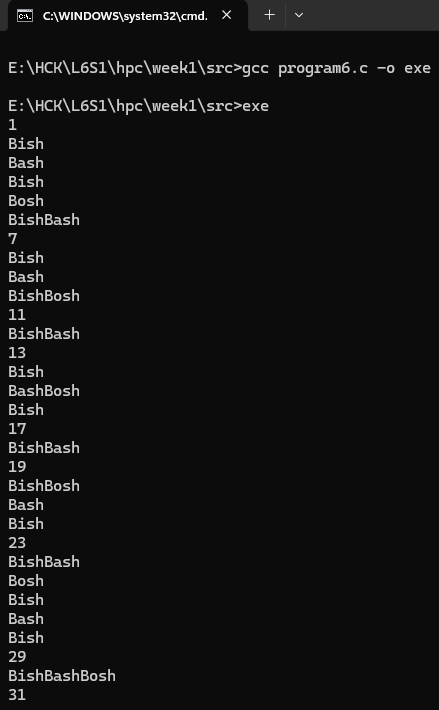
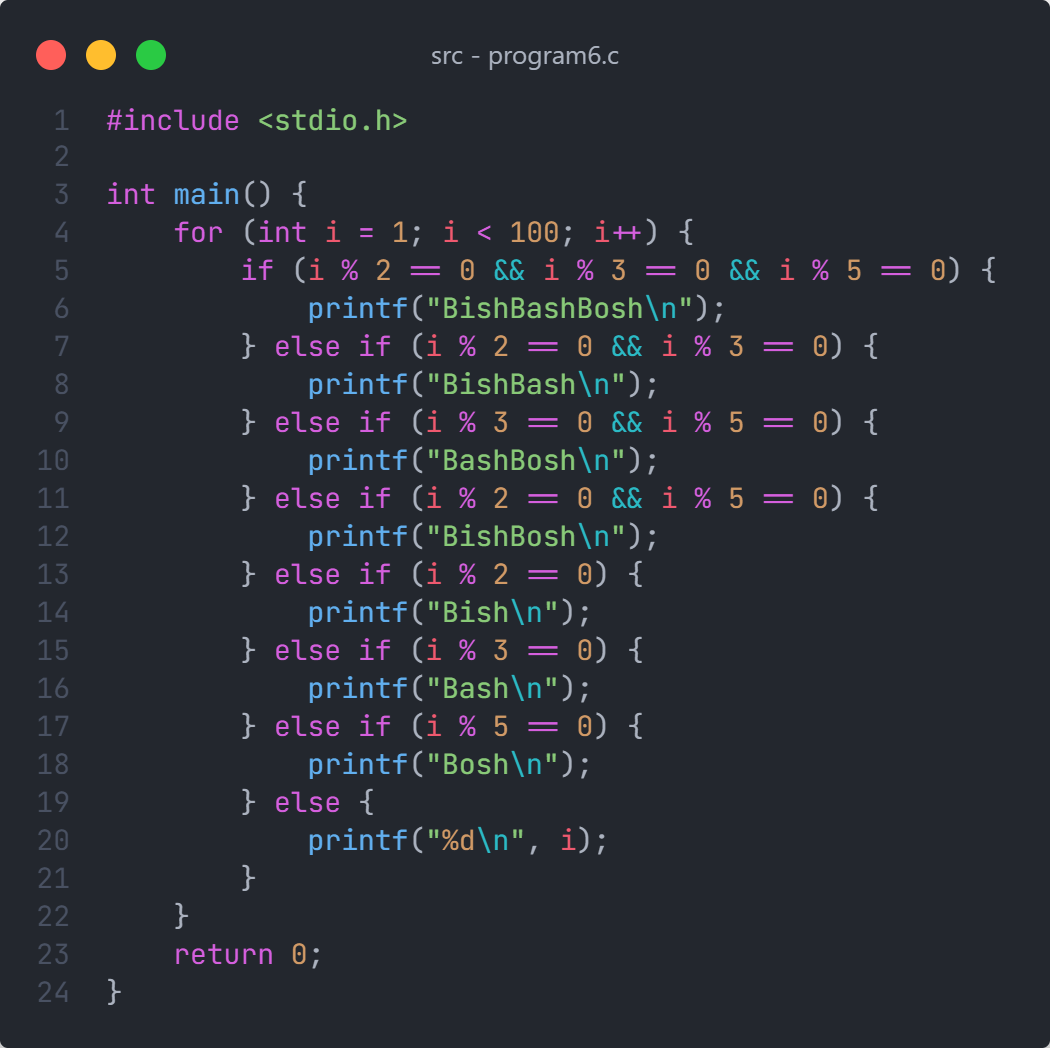


Figure 14: Source Code of Program 6



The following code swaps the values of the two variables a and b ::

#include <stdio.h>

void main(int argc, char \*argv[])

{

int a = 3;

int b = 4;

int temp = 0;

printf("a is %d and b is %d\n", a, b);

temp = a;

a = b;

b = temp;

printf("a is now %d and b is now %d", a, b);

}

7. Now write a function called swap() that would swap the values of the variables a and b, when you call the swap() with the variables a and b as parameters. Please note, this exercise requires pointers.



Figure 15: Source Code of Program 7

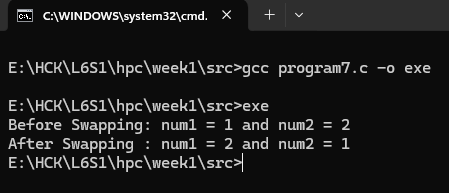


Figure 16: Output of program 7

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

#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();

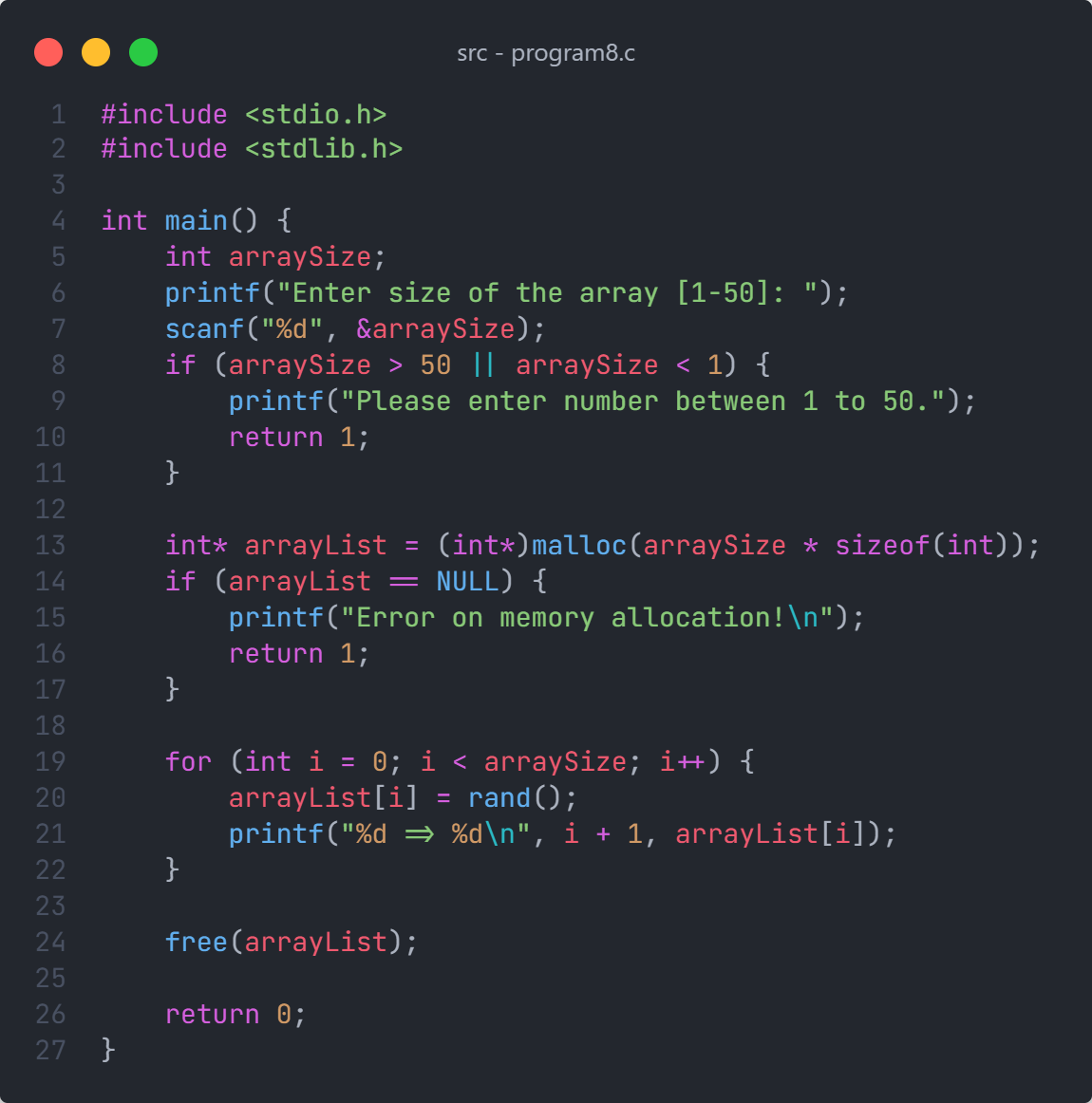
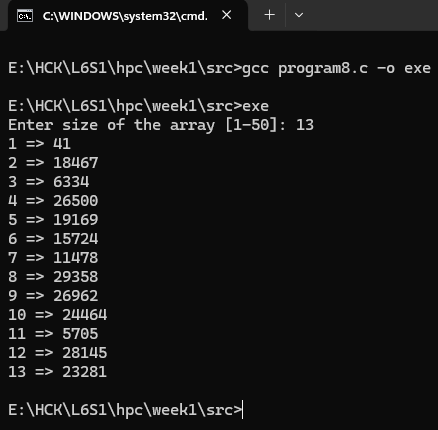
printf("%d is %d\n", i, numbers[i]);

}

}

8. 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.

Figure 17: Output of program 8



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

Figure 18: Source Code of program 8

#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);

}

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

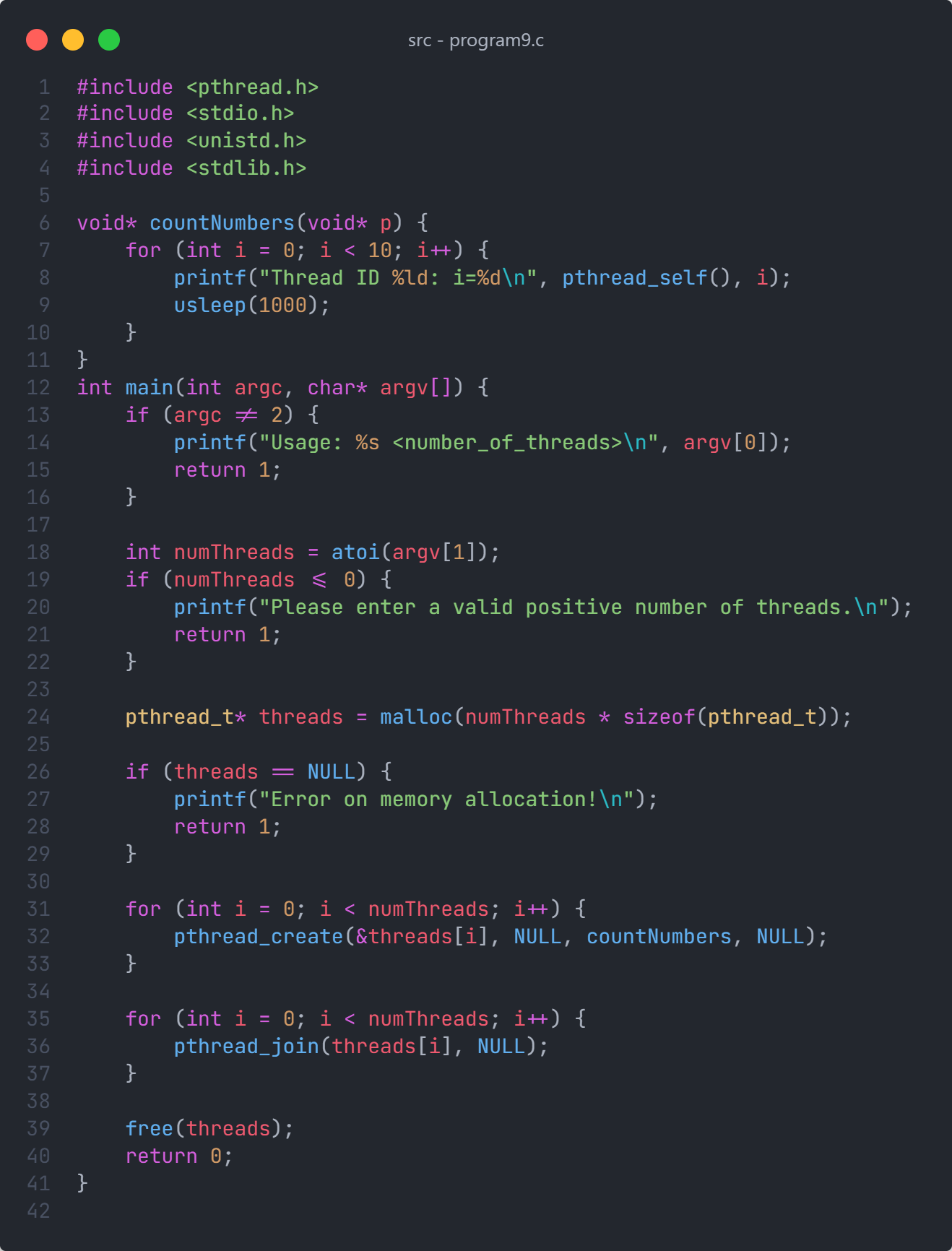


Figure 19: Source code of program 9

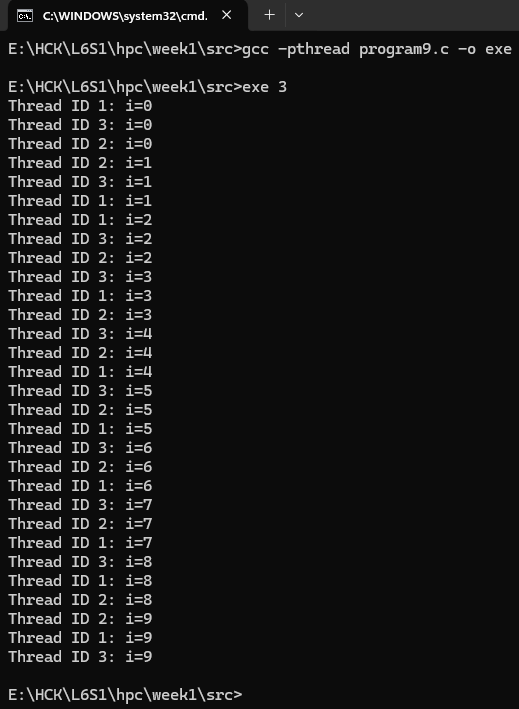


Figure 20: Output of program 9