NITHIN S
221IT085

CS111 Lab Assignment 9

Files

Q1) Test the size of structures, nested structures, unions, nested unions and justify the same.
Test as many cases as possible.

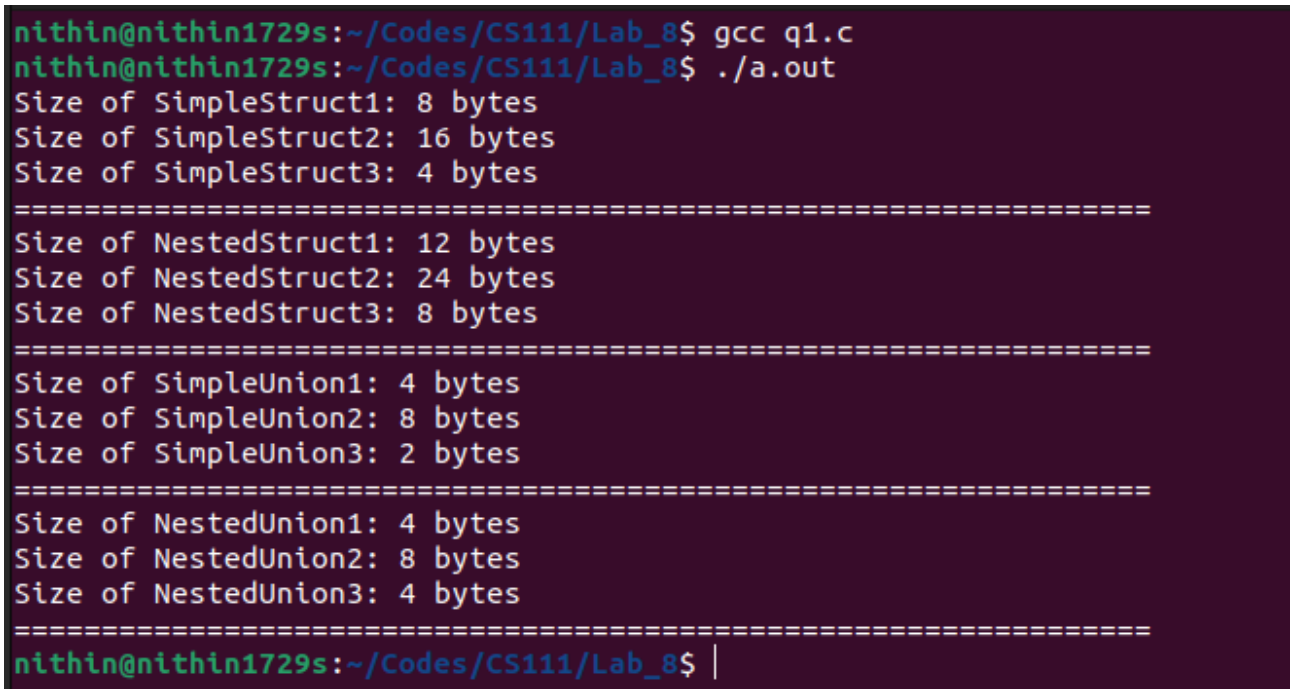
```
#include <stdio.h>


struct SimpleStruct1 {
    int a;
    char b;
};

struct SimpleStruct2 {
    double x;
    int y;
};

struct SimpleStruct3 {
    char c;
    short d;
};


struct NestedStruct1 {
    struct SimpleStruct1 inner1;
    float e;
};

struct NestedStruct2 {
```

```
    struct SimpleStruct2 inner2;
    char f;
};

struct NestedStruct3 {
    struct SimpleStruct3 inner3;
    int g;
};


union SimpleUnion1 {
    int h;
    char i;
};

union SimpleUnion2 {
    double j;
    int k;
};

union SimpleUnion3 {
    char l;
    short m;
};


union NestedUnion1 {
    union SimpleUnion1 inner4;
    float n;
};

union NestedUnion2 {
    union SimpleUnion2 inner5;
    char o;
};

union NestedUnion3 {
    union SimpleUnion3 inner6;
    int p;
};
```

```
int main() {

    printf("Size of SimpleStruct1: %lu bytes\n",
sizeof(struct SimpleStruct1));
    printf("Size of SimpleStruct2: %lu bytes\n",
sizeof(struct SimpleStruct2));
    printf("Size of SimpleStruct3: %lu bytes\n",
sizeof(struct SimpleStruct3));

printf("=============================================
====================\n");


    printf("Size of NestedStruct1: %lu bytes\n",
sizeof(struct NestedStruct1));
    printf("Size of NestedStruct2: %lu bytes\n",
sizeof(struct NestedStruct2));
    printf("Size of NestedStruct3: %lu bytes\n",
sizeof(struct NestedStruct3));

printf("=============================================
====================\n");



    printf("Size of SimpleUnion1: %lu bytes\n",
sizeof(union SimpleUnion1));
    printf("Size of SimpleUnion2: %lu bytes\n",
sizeof(union SimpleUnion2));
    printf("Size of SimpleUnion3: %lu bytes\n",
sizeof(union SimpleUnion3));

printf("=============================================
====================\n");



    printf("Size of NestedUnion1: %lu bytes\n",
sizeof(union NestedUnion1));
    printf("Size of NestedUnion2: %lu bytes\n",
sizeof(union NestedUnion2));
```

```
    printf("Size of NestedUnion3: %lu bytes\n",
sizeof(union NestedUnion3));

printf("==========================================
==========\n");


    return 0;
}
```

OUTPUT



Justification:

The sizes of structures and unions in C are determined by the compiler and its alignment requirements. The compiler tries to align data structures in memory to improve access speed, and padding may be introduced to achieve this alignment.

Simple Structures

SimpleStruct1:
   - `int` typically requires alignment on a 4-byte boundary.
   - `char` usually doesn't require alignment.
   - Possible padding to align the `char` with the next 4-byte boundary.
   - Total size might be 8 bytes (4 for `int` + 1 for `char` + 3 padding).

SimpleStruct2:
   - `double` often requires alignment on an 8-byte boundary.
   - `int` typically aligns on a 4-byte boundary.
   - Possible padding to align the `int` with the next 8-byte boundary.
   - Total size might be 16 bytes (8 for `double` + 4 for `int` + 4 padding).

SimpleStruct3:
   - `char` usually doesn't require alignment.
   - `short` might align on a 2-byte boundary.
   - Possible padding to align the `short` with the next 4-byte boundary.
   - Total size might be 4 bytes (1 for `char` + 2 for `short` + 1 padding).

Nested Structures

NestedStruct1:
   - Contains `SimpleStruct1` and a `float`.
   - Size of `SimpleStruct1` (8 bytes) + `float` (4 bytes) might result in 12 bytes.

NestedStruct2:
   - Contains `SimpleStruct2` and a `char`.
   - Size of `SimpleStruct2` (16 bytes) + `char` (1 byte) might result in 20 bytes.

NestedStruct3:
   - Contains `SimpleStruct3` and an `int`.
   - Size of `SimpleStruct3` (4 bytes) + `int` (4 bytes) might result in 8 bytes.

Simple Unions

SimpleUnion1:

   - Size is the maximum size of its members.
   - Either `int` (4 bytes) or `char` (1 byte), resulting in 4 bytes.

SimpleUnion2:

   - Size is the maximum size of its members.
   - Either `double` (8 bytes) or `int` (4 bytes), resulting in 8 bytes.

SimpleUnion3:

   - Size is the maximum size of its members.
   - Either `char` (1 byte) or `short` (2 bytes), resulting in 2 bytes.

Nested Unions

NestedUnion1:

   - Contains `SimpleUnion1` and a `float`.
   - Size is the maximum size of `SimpleUnion1` (4 bytes) and `float` (4 bytes), resulting in 8 bytes.

NestedUnion2:

   - Contains `SimpleUnion2` and a `char`.
   - Size is the maximum size of `SimpleUnion2` (8 bytes) and `char` (1 byte), resulting in 8 bytes.

NestedUnion3:

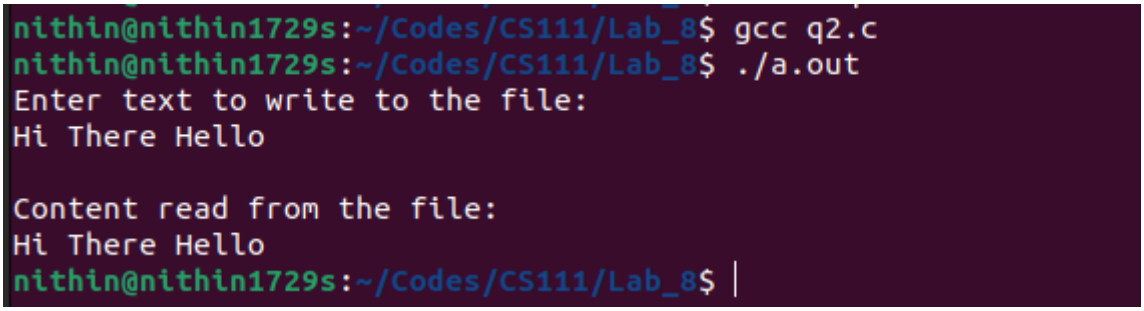
   - Contains `SimpleUnion3` and an `int`.
- Size is the maximum size of `SimpleUnion3` (2 bytes) and `int` (4 bytes), resulting in 4 bytes.

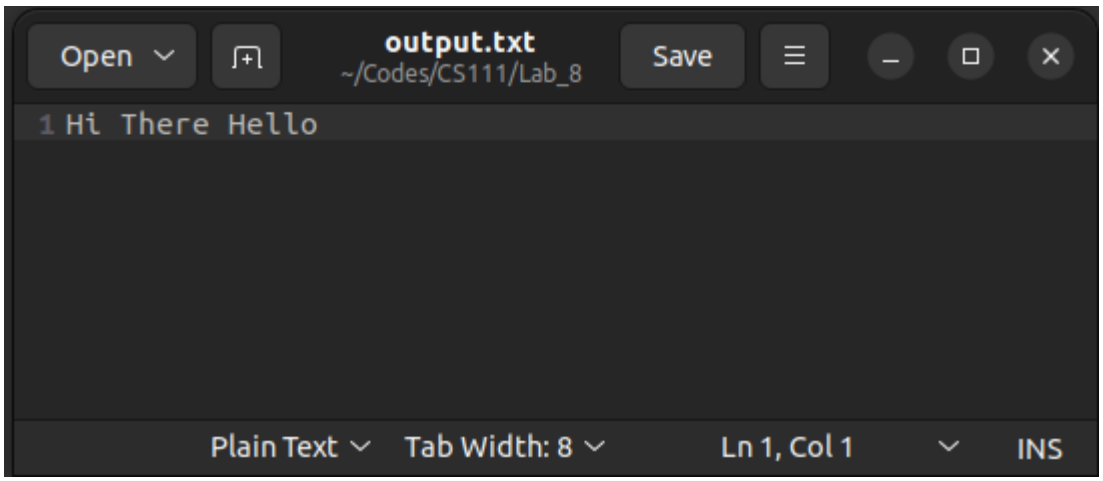
Q2) Write a C program to read and write from a File.

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

int main() {

    FILE *file;


    file = fopen("output.txt", "w");


    if (file == NULL) {
        perror("Error opening file for writing");
        return 1;
    }


    printf("Enter text to write to the file:\n");
    char input[100];
    fgets(input, sizeof(input), stdin);


    fprintf(file, "%s", input);


    fclose(file);


    file = fopen("output.txt", "r");


    if (file == NULL) {
        perror("Error opening file for reading");
```

```
        return 1;
    }


    printf("\nContent read from the file:\n");
    char buffer[100];
    while (fgets(buffer, sizeof(buffer), file) !=
NULL) {
        printf("%s", buffer);
    }


    fclose(file);

    return 0;
}
```

OUTPUT





Q3) Write a C program to write two integer values to file (new.txt) and read those values from file and perform addition on those two values and write the result back to the file(new.txt) without overwriting. The result must be displayed on the console output.

```
#include <stdio.h>

int main() {

    FILE *file;


    file = fopen("new.txt", "w");


    if (file == NULL) {
        perror("Error opening file for writing");
        return 1;
    }


    int value1, value2;
    printf("Enter the first number: ");
    scanf("%d", &value1);
    printf("Enter the second number: ");
    scanf("%d", &value2);


    fprintf(file, "%d %d", value1, value2);


    fclose(file);


    file = fopen("new.txt", "r");


    if (file == NULL) {
        perror("Error opening file for reading");
        return 1;
```

```
    }


    fseek(file, 0, SEEK_SET);
    int readValue1, readValue2;
    fscanf(file, "%d %d", &readValue1, &readValue2);


    fclose(file);


    int result = readValue1 + readValue2;


    file = fopen("new.txt", "a");


    if (file == NULL) {
        perror("Error opening file for appending");
        return 1;
    }


    fprintf(file, "\nResult: %d", result);


    fclose(file);


    printf("Result: %d\n", result);

    return 0;
}
```

OUTPUT

```
nithin@nithin1729s:~/Codes/CS111/Lab_8$ gcc q3.c
nithin@nithin1729s:~/Codes/CS111/Lab_8$ ./a.out
Enter the first number: 54
Enter the second number: 17
Result: 71
nithin@nithin1729s:~/Codes/CS111/Lab_8$
```

Q4) Write a C program to read EmployeeNo, Name, Designation, Basic Salary and Grade of Employees and store this data in a file "emp.txt“. Display the employee list from this file

```
#include <stdio.h>


struct Employee {
    int employeeId;
    char name[50];
    float salary;
};

int main() {

    FILE *file;
    struct Employee emp;
```

```
    file = fopen("emp.txt", "w");


    if (file == NULL) {
        perror("Error opening file for writing");
        return 1;
    }


    int numEmployees;
    printf("Enter the number of employees: ");
    scanf("%d", &numEmployees);


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



        printf("\nEnter details for Employee %d:\n", i +
1);
        printf("Employee ID: ");
        scanf("%d", &emp.employeeId);

        printf("Name: ");
        scanf("%s", emp.name);

        printf("Salary: ");
        scanf("%f", &emp.salary);


        fprintf(file, "%d %s %.2f\n", emp.employeeId,
emp.name, emp.salary);
    }


    fclose(file);


    file = fopen("emp.txt", "r");


    if (file == NULL) {
```

```
        perror("Error opening file for reading");
        return 1;
    }


    printf("\nEmployee List:\n");
    while (fscanf(file, "%d %s %f", &emp.employeeId,
emp.name, &emp.salary) == 3) {
        printf("Employee ID: %d\n", emp.employeeId);
        printf("Name: %s\n", emp.name);
        printf("Salary: %.2f\n", emp.salary);
        printf("----------------------\n");
    }


    fclose(file);

    return 0;
}
```

OUTPUT

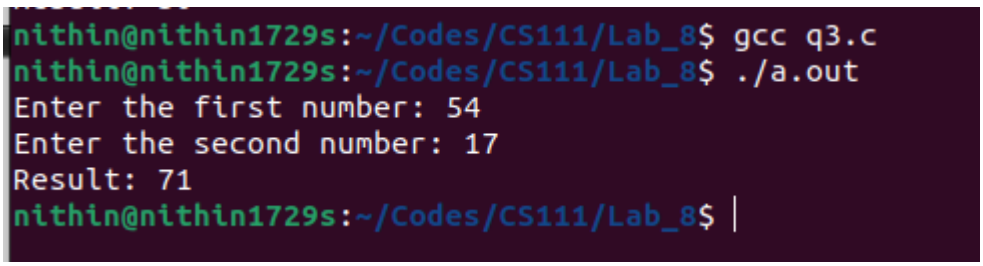
```
nithin@nithin1729s:~/Codes/CS111/Lab_8$ gcc q4.c
nithin@nithin1729s:~/Codes/CS111/Lab_8$ ./a.out
Enter the number of employees: 3

Enter details for Employee 1:
Employee ID: 45
Name: Nithin
Salary: 56000

Enter details for Employee 2:
Employee ID: 67
Name: Suraj
Salary: 67767776776

Enter details for Employee 3:
Employee ID: 45
Name: Bain
Salary: 78333

Employee List:
Employee ID: 45
Name: Nithin
Salary: 56000.00
--------------------
Employee ID: 67
Name: Suraj
Salary: 67767775232.00
--------------------
Employee ID: 45
Name: Bain
Salary: 78333.00
--------------------
```

Q5) Write a C program to write two integer values to file (new.txt) and read those values from file and perform addition on those two values and write the result back to the file(new.txt) without overwriting. The result must be displayed on the console output.

```
#include <stdio.h>

int main() {

    FILE *file;


    file = fopen("new.txt", "w");


    if (file == NULL) {
        perror("Error opening file for writing");
        return 1;
    }


    int value1, value2;
    printf("Enter the first number: ");
    scanf("%d", &value1);
    printf("Enter the second number: ");
    scanf("%d", &value2);


    fprintf(file, "%d %d", value1, value2);


    fclose(file);


    file = fopen("new.txt", "r");


    if (file == NULL) {
        perror("Error opening file for reading");
        return 1;
    }
```

```
    fseek(file, 0, SEEK_SET);
    int readValue1, readValue2;
    fscanf(file, "%d %d", &readValue1, &readValue2);


    fclose(file);


    int result = readValue1 + readValue2;


    file = fopen("new.txt", "a");


    if (file == NULL) {
        perror("Error opening file for appending");
        return 1;
    }


    fprintf(file, "\nResult: %d", result);


    fclose(file);


    printf("Result: %d\n", result);

    return 0;
}
```

OUTPUT




Open
new.txt
~/Codes/CS111/Lab_8
Save
1 54 17
2 Result: 71
Loading ...
Plain Text
Tab Width: 8
Ln 1, Col 1
INS