

# **PATUAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY**

**COURSE CODE CIT-112**

## **SUBMITTED TO:**

**Md. Mahbubur Rahman**

**Department of Computer Science and Information  
Technology  
Faculty of Computer Science and Engineering**

## **SUBMITTED BY:**

**Md. Sharafat Karim**

**ID: 2102024,**

**Registration No: 10151**

**Faculty of Computer Science and Engineering**

**Assignment: 09**

## Table of Contents

11.1 Define a structure data type called time_struct containing three members integer hour, integer minute and integer second . Develop a program that would assign values to the individual members and display the time in the following form:.....	6
11.2 Modify the above program such that a function is used to input values to the members and another function to display the time.....	7
11.3 Design a function update that would accept the data structure designed in Exercise 11.1 and increments time by one second and returns the new time. (If the increment results in 60 seconds, then the second member is set to zero and the minute member is incremented by one. Then, if the result is 60 minutes, the minute member is set to zero and the hour member is incremented by one. Finally when the hour becomes 24, it is set to zero.).....	8
11.4 Define a structure data type named date containing three integer members day , month , and year. Develop an interactive modular program to perform the following tasks:.....	10
11.5 Design a function update that accepts the date structure designed in Exercise 11.4 to increment the date by one day and return the new date. The following rules are applicable:.....	12
11.6 Modify the input function used in Exercise 11.4 such that it reads a value that represents the date in the form of a long integer, like 19450815 for the date 15-8-1945 (August 15, 1945) and assigns suitable values to the members day, month , and year.....	16
11.7 Add a function called nextdate to the program designed in Exercise 11.4 to perform the following task:.....	18
11.8 Use the date structure defined in Exercise 11.4 to store two dates. Develop a function that will take these two dates as input and compares them.....	22
11.9 Define a structure to represent a vector (a series of integer values) and write a modular program to perform the following tasks:.....	26

11.10 Add a function to the program of Exercise 11.9 that accepts two vectors as input parameters and return the addition of two vectors.....	29
11.11 Create two structures named metric and British which store the values of distances. The metric structure stores the values in metres and centimetres and the British structure stores the values in feet and inches. Write a program that reads values for the structure variables and adds values contained in one variable of metric to the contents of another variable of British. The program should display the result in the format of feet and inches or metres and centimetres as required.....	32
11.12 Define a structure named census with the following three members:.....	34
11.13 Define a structure that can describe an hotel. It should have members that include the name, address, grade, average room charge, and number of rooms. Write functions to perform the following operations:.....	38
11.14 Define a structure called cricket that will describe the following information:.....	42
11.15 Design a structure student_record to contain name, date of birth, and total marks obtained. Use the date structure designed in Exercise 11.4 to represent the date of birth. Develop a program to read data for 10 students in a class and list them rank-wise.....	45
11.16 Write a C program that prints the size of a structure data type.....	48
11.17 Write a C program that prints the size of a structure and union data type that have same number and type of members.....	49
11.18 Write a C program for demonstrating operations on individual structure members using pointer notation.....	50
19 Write a program to copy the contents of one file into another.....	52
20 Two files DATA1 and DATA2 contain sorted lists of integers. Write a program to produce a third file DATA which holds a single sorted, merged list of these two lists. Use command line arguments to specify the file names.....	53
21 Write a program that compares two files and returns 0 if they are equal and 1 is they are not.....	55

22 Write a program that appends one file at the end of another.....	56
23 Write a program that reads a file containing integers and appends at its end the sum of all the integers.....	57
24 Write a program that prompts the user for two files, one containing a line of text known as source file and other, an empty file known as target file and then copies the contents of source file into target file. Modify the program so that a specified character is deleted from the source file as it is copied to the target file.	58
25 Write a program that requests for a file name and an integer, known as offset value. The program then reads the file starting from the location specified by the offset value and prints the contents on the screen. Note: If the offset value is a positive integer, then printing skips that many lines. If it is a negative number, it prints that many lines from the end of the file. An appropriate error message should be printed, if anything goes wrong.....	59
26 Write a program to create a sequential file that could store details about five products. Details include product code, cost and number of items available and are provided through keyboard.....	62
27 Write a program to read the file created in Exercise 13.8 and compute and print the total value of all the five products.....	63
28 Rewrite the program developed in Exercise 13.8 to store the details in a random access file and print the details of alternate products from the file. Modify the program so that it can output the details of a product when its code is specified interactively.....	64
29 Write a C program that uses file handling methods to store records of mixed data in a file.....	67
30 Write a C program that uses getw function to read integer values from one file. Subsequently, it uses the putw function to write the integer values in reverse order in another file.....	69
31 Write a C program that reads characters from a file and prints their ASCII codes.....	71
32 Write a C program that concatenates the contents of two files and writes then in the third file.....	72

33 Write a C program that uses fscanf function to read integer values from a file, computes the square of each integer value and places the resultant values in a different file..... 73

**11.1 Define a structure data type called *time\_struct* containing three members integer *hour*, integer *minute* and integer *second* . Develop a program that would assign values to the individual members and display the time in the following form:**

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

```
struct time_struct
```

```
{
```

```
    int hour;
```

```
    int minute;
```

```
    int second;
```

```
};
```

```
int main()
```

```
{
```

```
    struct time_struct time;
```

```
    printf("Enter time in HH:MM:SS format: ");
```

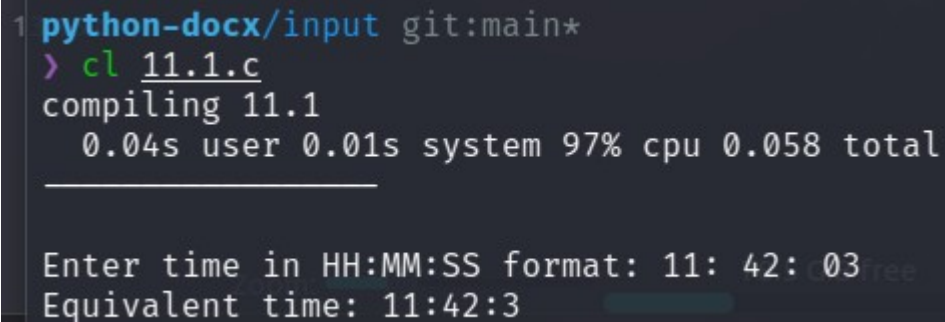
```
    scanf("%d:%d:%d", &time.hour, &time.minute, &time.second);
```

```
    printf("Equivalent time: ");
```

```
    printf("%d:%d:%d\n", time.hour, time.minute, time.second);
```

```
    return 0;
```

```
}
```



```
1 python-docx/input git:main*
> cl 11.1.c
compiling 11.1
 0.04s user 0.01s system 97% cpu 0.058 total

Enter time in HH:MM:SS format: 11: 42: 03
Equivalent time: 11:42:3
```

## 11.2 Modify the above program such that a function is used to input values to the members and another function to display the time.

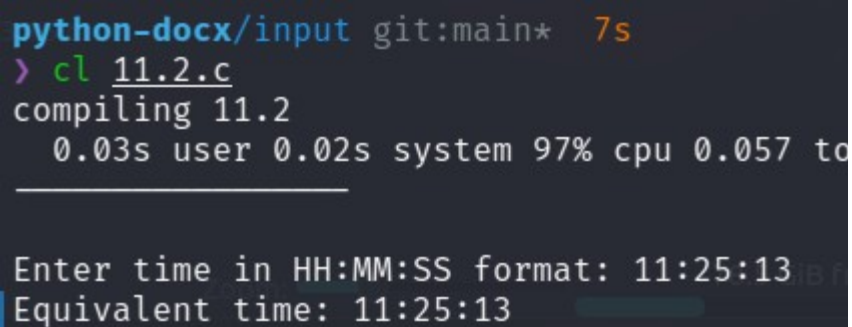
```
#include <stdio.h>

struct time_struct
{
    int hour;
    int minute;
    int second;
};

void input_time(struct time_struct *time)
{
    printf("Enter time in HH:MM:SS format: ");
    scanf("%d:%d:%d", &time->hour, &time->minute, &time->second);
}

void print_time(struct time_struct time)
{
    printf("Equivalent time: ");
    printf("%d:%d:%d\n", time.hour, time.minute, time.second);
}

int main()
{
    struct time_struct time;
    input_time(&time);
    print_time(time);
    return 0;
}
```



```
python-docx/input git:main* 7s
> cl 11.2.c
compiling 11.2
0.03s user 0.02s system 97% cpu 0.057 to

Enter time in HH:MM:SS format: 11:25:13
Equivalent time: 11:25:13
```

**11.3** Design a function *update* that would accept the data structure designed in Exercise 11.1 and increments time by one second and returns the new time. (If the increment results in 60 seconds, then the second member is set to zero and the minute member is incremented by one. Then, if the result is 60 minutes, the minute member is set to zero and the hour member is incremented by one. Finally when the hour becomes 24, it is set to zero.)

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

```
struct time_struct  
{  
    int hour;  
    int minute;  
    int second;  
};
```

```
void input_time(struct time_struct *time)  
{  
    printf("Enter time in HH:MM:SS format: ");  
    scanf("%d:%d:%d", &time->hour, &time->minute, &time->second);  
}
```

```
void print_time(struct time_struct time)  
{  
    printf("Equivalent time: ");  
    printf("%d:%d:%d\n", time.hour, time.minute, time.second);  
}
```

```
struct time_struct increment_second_in_time(struct time_struct time)  
{  
    time.second++;  
    if (time.second == 60)  
    {
```



```

    time.second = 0;
    time.minute++;
    if (time.minute == 60)
    {
        time.minute = 0;
        time.hour++;
        if (time.hour == 24)
        {
            time.hour = 0;
        }
    }
}
return time;
}

int main()
{
    struct time_struct time;
    input_time(&time);
    print_time(time);
    printf("Time after incrementing second: ");
    time = increment_second_in_time(time);
    print_time(time);
    return 0;
}

```

```

python-docx/input git:main* 8s
> cl 11.3.c
compiling 11.3
  0.04s user 0.02s system 95% cpu 0.062 total

Enter time in HH:MM:SS format: 12:59:59
Equivalent time: 12:59:59
Time after incrementing second: Equivalent time: 13:0:0

```

**11.4 Define a structure data type named *date* containing three integer members *day* , *month* , and *year*. Develop an interactive modular program to perform the following tasks:**

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

```
struct date
{
    int day;
    int month;
    int year;
};
```

```
void input_date(struct date *date)
{
    printf("Enter date in DD/MM/YYYY format: ");
    scanf("%d/%d/%d", &date->day, &date->month, &date->year);
}
```

```
void validate_data(struct date *date)
{
    if (date->year < 1)
    {
        printf("Invalid year\n");
        exit(1);
    }
    if (date->month < 1 || date->month > 12)
    {
        printf("Invalid month\n");
        exit(1);
    }
    if (((date->day < 1 || date->day > 31) || (date->day > 30 && (date->month == 4 ||
date->month == 6 || date->month == 9 || date->month == 11)) || (date->day > 28
&& date->month == 2) || (date->day > 29 && date->month == 2 && (date->year %
```

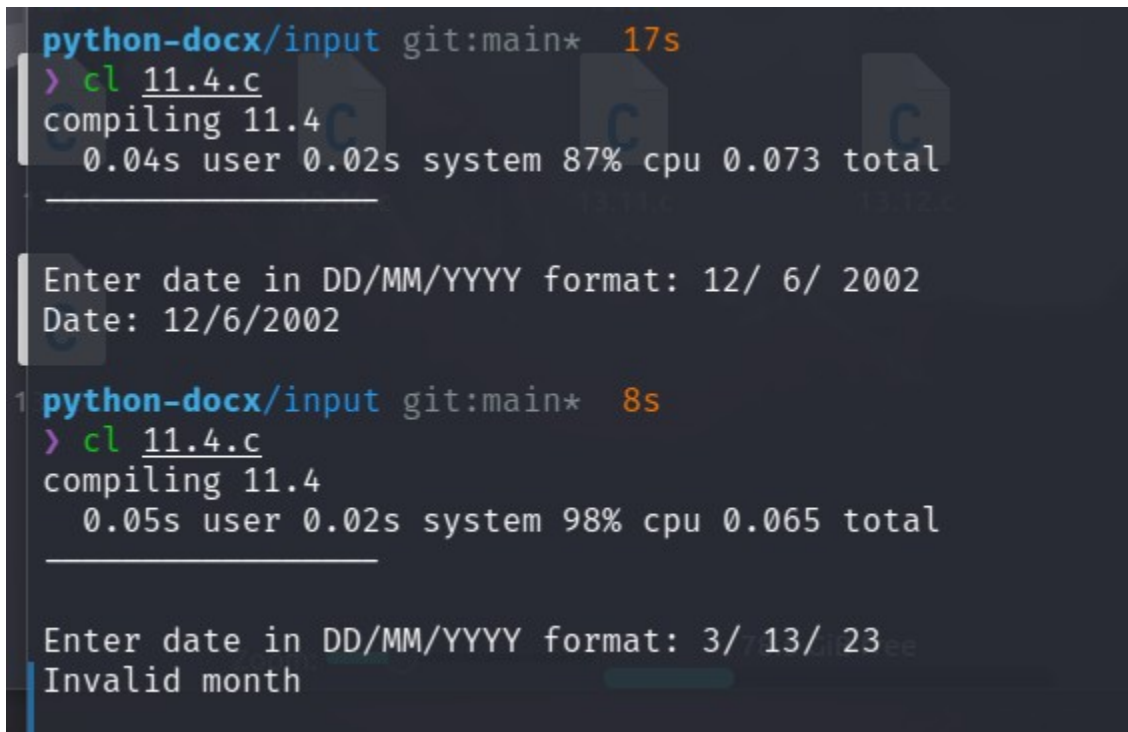
```

4 != 0 || (date->year % 100 == 0 && date->year % 400 != 0))))
{
    printf("Invalid day\n");
    exit(1);
}
}

void print_date(struct date *date)
{
    printf("Date: %d/%d/%d\n", date->day, date->month, date->year);
}

int main()
{
    struct date date;
    input_date(&date);
    validate_data(&date);
    print_date(&date);
    return 0;
}

```



The image shows a terminal window with two sessions. The first session shows the compilation of `11.4.c` into `11.4` with a success message. The program is then run, prompting the user to enter a date in DD/MM/YYYY format. The user enters `12/ 6/ 2002`, and the program outputs `Date: 12/6/2002`. The second session shows the compilation of `11.4.c` into `11.4` with a success message. The program is then run, prompting the user to enter a date in DD/MM/YYYY format. The user enters `3/ 13/ 23`, and the program outputs `Invalid month`.

```

python-docx/input git:main* 17s
> cl 11.4.c
compiling 11.4
0.04s user 0.02s system 87% cpu 0.073 total

Enter date in DD/MM/YYYY format: 12/ 6/ 2002
Date: 12/6/2002

python-docx/input git:main* 8s
> cl 11.4.c
compiling 11.4
0.05s user 0.02s system 98% cpu 0.065 total

Enter date in DD/MM/YYYY format: 3/ 13/ 23
Invalid month

```

**11.5 Design a function *update* that accepts the *date* structure designed in Exercise 11.4 to increment the date by one day and return the new date. The following rules are applicable:**

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

struct date
{
    int day;
    int month;
    int year;
};

void input_date(struct date *date)
{
    printf("Enter date in DD/MM/YYYY format: ");
    scanf("%d/%d/%d", &date->day, &date->month, &date->year);
}

void validate_data(struct date *date)
{
    if (date->year < 1)
    {
        printf("Invalid year\n");
        exit(1);
    }
    if (date->month < 1 || date->month > 12)
    {
        printf("Invalid month\n");
        exit(1);
    }
    if (((date->day < 1 || date->day > 31) || (date->day > 30 && (date->month == 4 ||
date->month == 6 || date->month == 9 || date->month == 11)) || (date->day > 28
&& date->month == 2) || (date->day > 29 && date->month == 2 && (date->year %
```

```
4 != 0 || (date->year % 100 == 0 && date->year % 400 != 0))))
```

```
{  
    printf("Invalid day\n");  
    exit(1);  
}  
}
```

```
int validate_data_and_return(struct date date)
```

```
{  
    if (date.year < 1)  
    {  
        // printf("Invalid year\n");  
        return(1);  
    }  
    if (date.month < 1 || date.month > 12)  
    {  
        // printf("Invalid month\n");  
        return(1);  
    }  
    if (((date.day < 1 || date.day > 31) || (date.day > 30 && (date.month == 4 ||  
date.month == 6 || date.month == 9 || date.month == 11)) || (date.day > 28 &&  
date.month == 2) || (date.day > 29 && date.month == 2 && (date.year % 4 != 0 ||  
(date.year % 100 == 0 && date.year % 400 != 0))))  
    {  
        // printf("Invalid day\n");  
        return(1);  
    }  
    return(0);  
}
```

```
void update(struct date *date)
```

```
{  
    struct date temp_date = *date;  
    temp_date.day++;  
    if (validate_data_and_return(temp_date))  
    {  
        temp_date.day = 1;  
    }
```

```

    temp_date.month++;
    if (validate_data_and_return(temp_date))
    {
        temp_date.month = 1;
        temp_date.year++;
        if (validate_data_and_return(temp_date))
        {
            printf("Invalid date\n");
            exit(1);
        }
    }
}
*date = temp_date;
}

void print_date(struct date *date)
{
    printf("Date: %d/%d/%d\n", date->day, date->month, date->year);
}

int main()
{
    struct date date;
    input_date(&date);
    validate_data(&date);
    print_date(&date);

    printf("Date after incrementing day: \n");
    update(&date);
    print_date(&date);
    return 0;
}

```

python-docx/input git:main\* 388s

> cl 11.5.c

compiling 11.5

0.07s user 0.03s system 91% cpu 0.101 total

---

Enter date in DD/MM/YYYY format: 12/3/2023

Date: 12/3/2023

Date after incrementing day:

Date: 13/3/2023

python-docx/input git:main\* 8s

> cl 11.5.c

compiling 11.5

0.05s user 0.02s system 96% cpu 0.068 total

---

Enter date in DD/MM/YYYY format: 31/12/2023

Date: 31/12/2023

Date after incrementing day:

Date: 1/1/2024

78.5 GiB free

**11.6 Modify the input function used in Exercise 11.4 such that it reads a value that represents the date in the form of a long integer, like 19450815 for the date 15-8-1945 (August 15, 1945) and assigns suitable values to the members day, month , and year.**

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

```
struct date
{
    int day;
    int month;
    int year;
};
```

```
void input_date(struct date *date)
{
    printf("Enter date in YYYYMMDD format: ");
    scanf("%4d%2d%2d", &date->year, &date->month, &date->day);
}
```

```
void validate_data(struct date *date)
{
    if (date->year < 1)
    {
        printf("Invalid year\n");
        exit(1);
    }
    if (date->month < 1 || date->month > 12)
    {
        printf("Invalid month\n");
        exit(1);
    }
    if (((date->day < 1 || date->day > 31) || (date->day > 30 && (date->month == 4 ||
date->month == 6 || date->month == 9 || date->month == 11)) || (date->day > 28
```



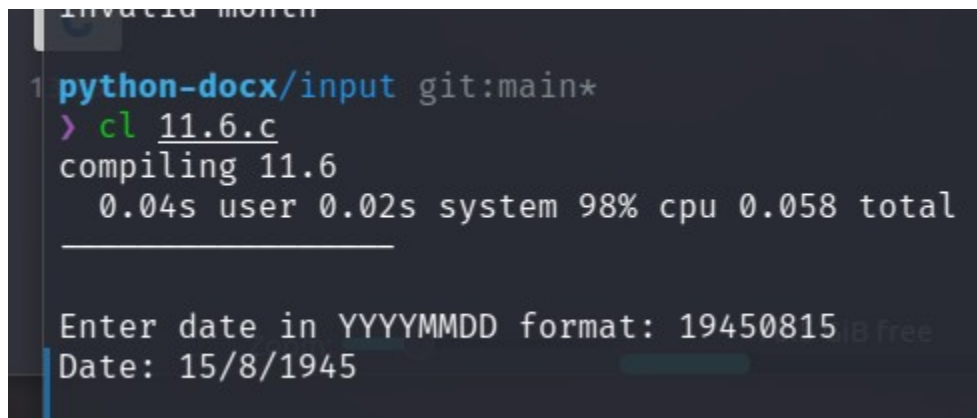
```

    && date->month == 2) || (date->day > 29 && date->month == 2 && (date->year %
4 != 0 || (date->year % 100 == 0 && date->year % 400 != 0))))
    {
        printf("Invalid day\n");
        exit(1);
    }
}

void print_date(struct date *date)
{
    printf("Date: %d/%d/%d\n", date->day, date->month, date->year);
}

int main()
{
    struct date date;
    input_date(&date);
    validate_data(&date);
    print_date(&date);
    return 0;
}

```



```

python-docx/input git:main*
> cl 11.6.c
compiling 11.6
0.04s user 0.02s system 98% cpu 0.058 total

Enter date in YYYYMMDD format: 19450815
Date: 15/8/1945

```

**11.7 Add a function called nextdate to the program designed in Exercise 11.4 to perform the following task:**

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

struct date
{
    int day;
    int month;
    int year;
};

void input_date(struct date *date)
{
    printf("Enter date in DD/MM/YYYY format: ");
    scanf("%d/%d/%d", &date->day, &date->month, &date->year);
}

void validate_data(struct date *date)
{
    if (date->year < 1)
    {
        printf("Invalid year\n");
        exit(1);
    }
    if (date->month < 1 || date->month > 12)
    {
        printf("Invalid month\n");
        exit(1);
    }
    if (((date->day < 1 || date->day > 31) || (date->day > 30 && (date->month == 4 ||
date->month == 6 || date->month == 9 || date->month == 11)) || (date->day > 28
&& date->month == 2) || (date->day > 29 && date->month == 2 && (date->year %
4 != 0 || (date->year % 100 == 0 && date->year % 400 != 0))))))
```

```

    {
        printf("Invalid day\n");
        exit(1);
    }
}

```

```

int validate_data_and_return(struct date date)

```

```

{
    if (date.year < 1)
    {
        // printf("Invalid year\n");
        return(1);
    }
    if (date.month < 1 || date.month > 12)
    {
        // printf("Invalid month\n");
        return(1);
    }
    if ((date.day < 1 || date.day > 31) || (date.day > 30 && (date.month == 4 ||
date.month == 6 || date.month == 9 || date.month == 11)) || (date.day > 28 &&
date.month == 2) || (date.day > 29 && date.month == 2 && (date.year % 4 != 0 ||
(date.year % 100 == 0 && date.year % 400 != 0))))
    {
        // printf("Invalid day\n");
        return(1);
    }
    return(0);
}

```

```

void update(struct date *date)

```

```

{
    struct date temp_date = *date;
    temp_date.day++;
    if (validate_data_and_return(temp_date))
    {
        temp_date.day = 1;
        temp_date.month++;
    }
}

```

```

    if (validate_data_and_return(temp_date))
    {
        temp_date.month = 1;
        temp_date.year++;
        if (validate_data_and_return(temp_date))
        {
            printf("Invalid date\n");
            exit(1);
        }
    }
}
*date = temp_date;
}

```

```

void nextdate(struct date *date)
{
    int days_to_add = 1;
    printf("Enter number of days to add: ");
    scanf("%d", &days_to_add);
    for (int i = 0; i < days_to_add; i++)
    {
        update(date);
    }
}

```

```

void print_date(struct date *date)
{
    printf("Date: %d/%d/%d\n", date->day, date->month, date->year);
}

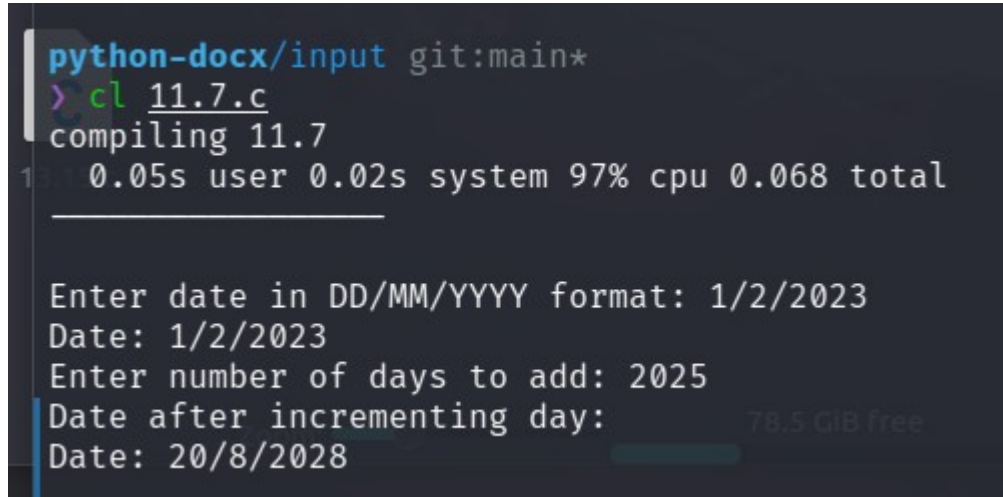
```

```

int main()
{
    struct date date;
    input_date(&date);
    validate_data(&date);
    print_date(&date);
}

```

```
nextdate(&date);  
printf("Date after incrementing day: \n");  
print_date(&date);  
return 0;  
}
```



```
python-docx/input git:main*  
> cl 11.7.c  
compiling 11.7  
1 0.05s user 0.02s system 97% cpu 0.068 total  
  
Enter date in DD/MM/YYYY format: 1/2/2023  
Date: 1/2/2023  
Enter number of days to add: 2025  
Date after incrementing day: 20/8/2028  
78.5 GiB free
```

The image shows a terminal window with a dark background. At the top, the prompt is 'python-docx/input git:main\*'. The user enters 'cl 11.7.c', and the system responds with 'compiling 11.7'. Below this, a line shows compilation statistics: '1 0.05s user 0.02s system 97% cpu 0.068 total'. A horizontal line separates this from the program's execution. The program prompts for a date in 'DD/MM/YYYY' format, and the user enters '1/2/2023'. The program then displays 'Date: 1/2/2023'. Next, it prompts for the number of days to add, and the user enters '2025'. Finally, the program displays 'Date after incrementing day: 20/8/2028'. In the bottom right corner of the terminal, the text '78.5 GiB free' is visible.

**11.8 Use the *date* structure defined in Exercise 11.4 to store two dates. Develop a function that will take these two dates as input and compares them.**

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

struct date
{
    int day;
    int month;
    int year;
};

void input_date(struct date *date)
{
    printf("Enter date in DD/MM/YYYY format: ");
    scanf("%d/%d/%d", &date->day, &date->month, &date->year);
}

void validate_data(struct date *date)
{
    if (date->year < 1)
    {
        printf("Invalid year\n");
        exit(1);
    }
    if (date->month < 1 || date->month > 12)
    {
        printf("Invalid month\n");
        exit(1);
    }
    if (((date->day < 1 || date->day > 31) || (date->day > 30 && (date->month == 4 ||
date->month == 6 || date->month == 9 || date->month == 11)) || (date->day > 28
&& date->month == 2) || (date->day > 29 && date->month == 2 && (date->year %
```

```
4 != 0 || (date->year % 100 == 0 && date->year % 400 != 0))))  
{  
    printf("Invalid day\n");  
    exit(1);  
}  
}
```

```
int compare_two_date(struct date *one, struct date *two)  
{  
    if (one->year > two->year)  
    {  
        return(1);  
    }  
    else if (one->year < two->year)  
    {  
        return(0);  
    }  
    else  
    {  
        if (one->month > two->month)  
        {  
            return(1);  
        }  
        else if (one->month < two->month)  
        {  
            return(0);  
        }  
        else  
        {  
            if (one->day > two->day)  
            {  
                return(1);  
            }  
            else if (one->day < two->day)  
            {  
                return(0);  
            }  
        }  
    }  
}
```

```
        else
        {
            return(0);
        }
    }
}
```

```
void print_date(struct date *date)
{
    printf("Date: %d/%d/%d\n", date->day, date->month, date->year);
}
```

```
int main()
{
    struct date date1, date2;
    input_date(&date1);
    validate_data(&date1);

    input_date(&date2);
    validate_data(&date2);

    if (!compare_two_date(&date1, &date2))
    {
        printf("Earlier date: ");
        print_date(&date1);
    }
    else
    {
        printf("Earlier date: ");
        print_date(&date2);
    }
    return 0;
}
```



```
python-docx/input git:main* 7s
```

```
> cl 11.8.c
```

```
compiling 11.8
```

```
0.05s user 0.02s system 88% cpu 0.082 total
```

---

```
Enter date in DD/MM/YYYY format: 1/3/2021
```

```
Enter date in DD/MM/YYYY format: 2/3/2021
```

```
Earlier date: Date: 1/3/2021
```

**11.9 Define a structure to represent a vector (a series of integer values) and write a modular program to perform the following tasks:**

```
#include <stdio.h>
#include <stdlib.h>
struct vector
{
    int *elements;
    int size;
};

void createVector(struct vector *v, int size)
{
    v->elements = (int *)malloc(size * sizeof(int));
    v -> size = size;
}

void addElement(struct vector *v)
{
    int element;
    for (int i = 0; i < v->size; i++)
    {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &element);
        v->elements[i] = element;
    }
}

void modifyElement(struct vector *v)
{
    int index;
    int newValue;
    printf("Enter index: ");
    scanf("%d", &index);
    printf("Enter new value: ");
```

```

        scanf("%d", &newValue);
        if (index >= 0 && index < v->size)
            v->elements[index] = newValue;
    }

void multiplyByScalar(struct vector *v)
{
    int scalar;
    printf("Enter scalar: ");
    scanf("%d", &scalar);
    for (int i = 0; i < v->size; i++)
    {
        v->elements[i] *= scalar;
    }
}

void displayVector(struct vector *v)
{
    printf("Vector: ");
    for (int i = 0; i < v->size; i++)
    {
        printf("%d ", v->elements[i]);
    }
    printf("\n");
}

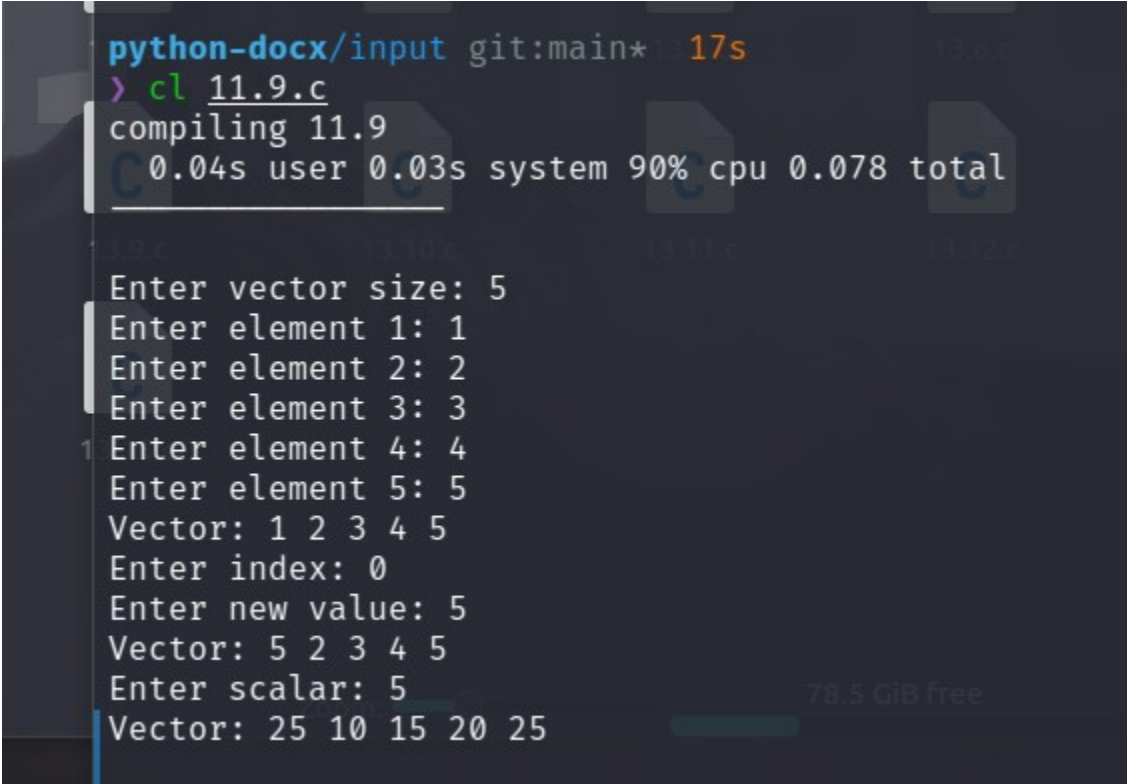
int main()
{
    struct vector v;
    int vector_size;
    printf("Enter vector size: ");
    scanf("%d", &vector_size);
    createVector(&v, vector_size);
    addElement(&v);
    displayVector(&v);

    modifyElement(&v);

```

```
displayVector(&v);

multiplyByScalar(&v);
displayVector(&v);
}
```

A terminal window with a dark background. The prompt is 'python-docx/input git:main\*17s'. The user enters 'cl 11.9.c', followed by 'compiling 11.9'. The output shows '0.04s user 0.03s system 90% cpu 0.078 total'. The user then enters 'Enter vector size: 5', followed by 'Enter element 1: 1', 'Enter element 2: 2', 'Enter element 3: 3', 'Enter element 4: 4', and 'Enter element 5: 5'. The program outputs 'Vector: 1 2 3 4 5'. The user enters 'Enter index: 0', followed by 'Enter new value: 5'. The program outputs 'Vector: 5 2 3 4 5'. The user enters 'Enter scalar: 5', and the program outputs 'Vector: 25 10 15 20 25'. In the bottom right corner, there is a status bar showing '78.5 GiB free' with a corresponding progress bar.

```
python-docx/input git:main*17s
> cl 11.9.c
compiling 11.9
0.04s user 0.03s system 90% cpu 0.078 total

Enter vector size: 5
Enter element 1: 1
Enter element 2: 2
Enter element 3: 3
Enter element 4: 4
Enter element 5: 5
Vector: 1 2 3 4 5
Enter index: 0
Enter new value: 5
Vector: 5 2 3 4 5
Enter scalar: 5
Vector: 25 10 15 20 25

78.5 GiB free
```

**11.10 Add a function to the program of Exercise 11.9 that accepts two vectors as input parameters and return the addition of two vectors.**

```
#include <stdio.h>
#include <stdlib.h>
struct vector
{
    int *elements;
    int size;
};

void createVector(struct vector *v, int size)
{
    v->elements = (int *)malloc(size * sizeof(int));
    v -> size = size;
}

void addElement(struct vector *v)
{
    int element;
    for (int i = 0; i < v->size; i++)
    {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &element);
        v->elements[i] = element;
    }
}

struct vector addition(struct vector *v1, struct vector *v2)
{
    struct vector v3;
    createVector(&v3, v1->size);
    for (int i = 0; i < v1->size; i++)
    {
        v3.elements[i] = v1->elements[i] + v2->elements[i];
    }
}
```

```
    }  
    return v3;  
}
```

```
void displayVector(struct vector *v)  
{  
    printf("Vector: ");  
    for (int i = 0; i < v->size; i++)  
    {  
        printf("%d ", v->elements[i]);  
    }  
    printf("\n");  
}
```

```
int main()  
{  
    struct vector v1, v2;  
    int vector_size;  
    printf("Enter vector size: ");  
    scanf("%d", &vector_size);  
  
    createVector(&v1, vector_size);  
    addElement(&v1);  
  
    createVector(&v2, vector_size);  
    addElement(&v2);  
  
    struct vector v3 = addition(&v1, &v2);  
    printf("Addition of two vectors: \n");  
    displayVector(&v3);  
}
```

python-docx/input git:main\* 10s

> cl 11.10.c

compiling 11.10

0.05s user 0.02s system 80% cpu 0.091 total

Enter vector size: 3

Enter element 1: 1

Enter element 2: 2

Enter element 3: 3

Enter element 1: 4

Enter element 2: 5

Enter element 3: 6

Addition of two vectors:

Vector: 5 7 9

78.5 GiB free

**11.11 Create two structures named *metric* and *British* which store the values of distances. The *metric* structure stores the values in metres and centimetres and the *British* structure stores the values in feet and inches. Write a program that reads values for the structure variables and adds values contained in one variable of *metric* to the contents of another variable of *British*. The program should display the result in the format of feet and inches or metres and centimetres as required.**

```
#include <stdio.h>

struct metric
{
    float meters;
    float centimeters;
};

struct British
{
    float feet;
    float inches;
};

int main()
{
    struct metric m;
    struct British b;

    printf("Enter the metric distance (meter): ");
    scanf("%f", &m.meters);
    printf("Enter the metric distance (centimeter): ");
    scanf("%f", &m.centimeters);

    b.feet = 3.28084 * m.meters;
    b.inches = 0.3937008 * m.centimeters;
```

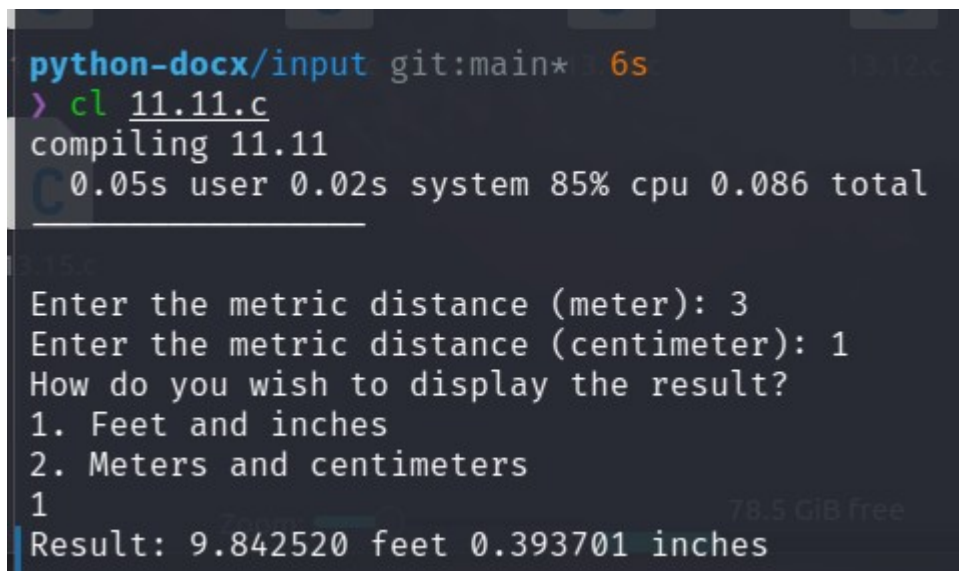


```

printf("How do you wish to display the result?\n");
printf("1. Feet and inches\n");
printf("2. Meters and centimeters\n");

int choice;
scanf("%d", &choice);
switch (choice)
{
    case 1:
        printf("Result: %f feet %f inches\n", b.feet, b.inches);
        break;
    case 2:
        printf("Result: %f meters %f centimeters\n", m.meters, m.centimeters);
        break;
    default:
        printf("Invalid choice\n");
}
return 0;
}

```



```

python-docx/input git:main* 6s 13.12.2
> cl 11.11.c
compiling 11.11
0.05s user 0.02s system 85% cpu 0.086 total

Enter the metric distance (meter): 3
Enter the metric distance (centimeter): 1
How do you wish to display the result?
1. Feet and inches
2. Meters and centimeters
1
Result: 9.842520 feet 0.393701 inches

```

### 11.12 Define a structure named *census* with the following three members:

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

struct census
{
    char city[20];
    long int population;
    float literacy;
};

void display_city(struct census city[])
{
    int i;
    printf("The details of the cities are:\n");
    printf(" City\t\tPopulation\tLiteracy rate\n");
    for (i = 0; i < 5; i++)
    {
        printf(" %s\t%d\t%f\n", city[i].city, city[i].population, city[i].literacy);
    }
}

void sort_list_alphabetically(struct census city[])
{
    int i, j;
    struct census temp;
    for (i = 0; i < 5; i++)
    {
        for (j = i; j < 5; j++)
        {
            if (strcmp(city[i].city, city[j].city) > 0)
            {
                temp = city[i];
                city[i] = city[j];
            }
        }
    }
}
```

```

        city[j] = temp;
    }
}
}
display_city(city);
}

```

```

void sort_list_by_population(struct census city[])
{
    int i, j;
    struct census temp;
    for (i = 0; i < 5; i++)
    {
        for (j = i; j < 5; j++)
        {
            if (city[i].population > city[j].population)
            {
                temp = city[i];
                city[i] = city[j];
                city[j] = temp;
            }
        }
    }
    display_city(city);
}

```

```

void sort_list_by_literacy(struct census city[])
{
    int i, j;
    struct census temp;
    for (i = 0; i < 5; i++)
    {
        for (j = i; j < 5; j++)
        {
            if (city[i].literacy > city[j].literacy)
            {
                temp = city[i];

```

```
        city[i] = city[j];
        city[j] = temp;
    }
}
}
display_city(city);
}
```

```
int main()
{
    struct census city[5];
    int i;
    for (i = 0; i < 5; i++)
    {
        printf("Enter the name of the city: ");
        scanf("%s", city[i].city);
        printf("Enter the population of the city: ");
        scanf("%ld", &city[i].population);
        printf("Enter the literacy rate of the city: ");
        scanf("%f", &city[i].literacy);
    }
    printf("\n");
    sort_list_alphabetically(city);

    printf("\n");
    sort_list_by_literacy(city);

    printf("\n");
    sort_list_by_population(city);

    return 0;
}
```

1. Feet and inches  
2. Meters and centimeters  
1  
Result: 9.842520 feet 0.393701 inches

python-dock/input git:main\* 7s

> 11.12.6

compiling 11.12

0.05s user 0.03s system 81% cpu 0.099 total

Enter the name of the city: dhaka

Enter the population of the city: 123

Enter the literacy rate of the city: 2

Enter the name of the city: edge

Enter the population of the city: 32

Enter the literacy rate of the city: 1

Enter the name of the city: sad

Enter the population of the city: 23

Enter the literacy rate of the city: 2

Enter the name of the city: df

Enter the population of the city: 212

Enter the literacy rate of the city: 21

Enter the name of the city: edd

Enter the population of the city: 1

Enter the literacy rate of the city: 2

The details of the cities are:

City	Population	Literacy rate
df	212	21.000000
dhaka	123	2.000000
edd	1	2.000000
edge	32	1.000000
sad	23	2.000000

The details of the cities are:

City	Population	Literacy rate
edge	32	1.000000
edd	1	2.000000
dhaka	123	2.000000
sad	23	2.000000
df	212	21.000000

The details of the cities are:

City	Population	Literacy rate
edd	1	2.000000
sad	23	2.000000
edge	32	1.000000
dhaka	123	2.000000
df	212	21.000000

python-dock/input git:main\* 25s

>

Network speed

Download: 0.0 B/s  
Upload: 0.0 B/s

Hard Disk Activity

Write Rate: 0.0 B/s  
Read Rate: 0.0 B/s

Individual Core Usage

Core 0: 1.2 B/s  
Core 1: 1.2 B/s

Memory Usage

44.0%

Free: 1.2 GB  
Total: 2.5 GB

**11.13 Define a structure that can describe an hotel. It should have members that include the name, address, grade, average room charge, and number of rooms. Write functions to perform the following operations:**

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

```
struct hotel
{
    char name[20];
    int address;
    int grade;
    int average_room_charge;
    int number_of_rooms;
};
```

```
void input_hotel(struct hotel *h, int number)
{
    int i;
    for (i = 0; i < number; i++)
    {
        printf("Enter the name of the hotel: ");
        scanf("%s", h[i].name);
        printf("Enter the address of the hotel: ");
        scanf("%d", &h[i].address);
        printf("Enter the grade of the hotel: ");
        scanf("%d", &h[i].grade);
        printf("Enter the average room charge of the hotel: ");
        scanf("%d", &h[i].average_room_charge);
        printf("Enter the number of rooms in the hotel: ");
        scanf("%d", &h[i].number_of_rooms);
        printf("\n");
    }
}
```

```

void sort_by_average_room_charge(struct hotel *h, int number)
{
    int i, j;
    struct hotel temp;
    for (i = 0; i < number; i++)
    {
        for (j = i; j < number; j++)
        {
            if (h[i].average_room_charge > h[j].average_room_charge)
            {
                temp = h[i];
                h[i] = h[j];
                h[j] = temp;
            }
        }
    }
}

```

```

void print_specific_grade(struct hotel *h, int number)
{
    int grade;
    printf("Enter the grade: ");
    scanf("%d", &grade);
    int i;
    for (i = 0; i < number; i++)
    {
        if (h[i].grade == grade)
        {
            printf("\tName: %s\n", h[i].name);
            printf("\tAddress: %d\n", h[i].address);
            printf("\tGrade: %d\n", h[i].grade);
            printf("\tAverage room charge: %d\n", h[i].average_room_charge);
            printf("\tNumber of rooms: %d\n", h[i].number_of_rooms);
            printf("\n");
        }
    }
}

```

```

void print_charge_less_than(struct hotel *h, int number)
{
    int charge;
    printf("Enter the max charge: ");
    scanf("%d", &charge);
    int i;
    for (i = 0; i < number; i++)
    {
        if (h[i].average_room_charge < charge)
        {
            printf("\tName: %s\n", h[i].name);
            printf("\tAddress: %d\n", h[i].address);
            printf("\tGrade: %d\n", h[i].grade);
            printf("\tAverage room charge: %d\n", h[i].average_room_charge);
            printf("\tNumber of rooms: %d\n", h[i].number_of_rooms);
        }
    }
}

int main()
{
    int n;
    printf("Enter the number of hotels: ");
    scanf("%d", &n);
    struct hotel h[n];
    input_hotel(h, n);

    sort_by_average_room_charge(h, n);
    print_specific_grade(h, n);

    print_charge_less_than(h, n);
}

```



```
input: zsh — Konsole
python-docx/input git:main* 25s
> cl 11.13.c
compiling 11.13
0.06s user 0.02s system 95% cpu 0.082 total

Enter the number of hotels: 2
Enter the name of the hotel: first
Enter the address of the hotel: 1353
Enter the grade of the hotel: 24
Enter the average room charge of the hotel: 45
Enter the number of rooms in the hotel: 45

Enter the name of the hotel: second
Enter the address of the hotel: 123
Enter the grade of the hotel: 542
Enter the average room charge of the hotel: 12
Enter the number of rooms in the hotel: 12

Enter the grade: 6
Enter the max charge: 35
    Name: second
    Address: 123
    Grade: 542
    Average room charge: 12
    Number of rooms: 12

python-docx/input git:main* 26s
> 
```

### 11.14 Define a structure called *cricket* that will describe the following information:

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

struct cricket
{
    char player_name[20];
    char team_name[20];
    int batting_average;
};

void input_player(struct cricket *p, int number)
{
    int i;
    for (i = 0; i < number; i++)
    {
        printf("Enter the name of the player: ");
        scanf("%s", p[i].player_name);
        printf("Enter the name of the team: ");
        scanf("%s", p[i].team_name);
        printf("Enter the batting average of the player: ");
        scanf("%d", &p[i].batting_average);
        printf("\n");
    }
}

void sort_by_team_name(struct cricket *p, int number)
{
    int i, j;
    struct cricket temp;
    for (i = 0; i < number; i++)
    {
        for (j = i; j < number; j++)
```

```

    {
        if (strcmp(p[i].team_name, p[j].team_name) > 0)
        {
            temp = p[i];
            p[i] = p[j];
            p[j] = temp;
        }
    }
}

```

```

void print_all_players(struct cricket *p, int number)

```

```

{
    int i;
    printf("All players:\n");
    printf("Player name\tTeam name\tBatting average\n");
    for (i = 0; i < number; i++)
    {
        printf("%s\t\t%s\t\t%d\n", p[i].player_name, p[i].team_name,
p[i].batting_average);
    }
}

```

```

int main()

```

```

{
    struct cricket player[50]; // make it a lower value to test
    input_player(player, 50);
    sort_by_team_name(player, 50);
    print_all_players(player, 50);
    return 0;
}

```

```
Enter the name of the team: 12
Enter the batting average of the player:
12
```

**11.15** Design a structure *student\_record* to contain name, date of birth, and total marks obtained. Use the *date* structure designed in Exercise 11.4 to represent the date of birth. Develop a program to read data for 10 students in a class and list them rank-wise.

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

struct date
{
    int day;
    int month;
    int year;
};

struct student_record
{
    char name[20];
    struct date date_of_birth;
    int total_marks;
};

void input_date(struct date *date)
{
    printf("Enter date in DD/MM/YYYY format: ");
    scanf("%d/%d/%d", &date->day, &date->month, &date->year);
}

void validate_data(struct date *date)
{
    if (date->year < 1)
    {
        printf("Invalid year\n");
        exit(1);
    }
}
```

```

if (date->month < 1 || date->month > 12)
{
    printf("Invalid month\n");
    exit(1);
}
if (((date->day < 1 || date->day > 31) || (date-> day > 30 && (date->month == 4 ||
date->month == 6 || date->month == 9 || date->month == 11)) || (date->day > 28
&& date->month == 2) || (date->day > 29 && date->month == 2 && (date->year %
4 != 0 || (date->year % 100 == 0 && date->year % 400 != 0))))
{
    printf("Invalid day\n");
    exit(1);
}
}

```

```

void print_date(struct date *date)
{
    printf("Date: %d/%d/%d\n", date->day, date->month, date->year);
}

```

```

void input_student_record(struct student_record *record)
{
    printf("Enter the name of the student: ");
    scanf("%s", record->name);
    input_date(&record->date_of_birth);
    validate_data(&record->date_of_birth);
    printf("Enter the total marks of the student: ");
    scanf("%d", &record->total_marks);
}

```

```

void print_student_record(struct student_record *record)
{
    printf("Name: %s\n", record->name);
    print_date(&record->date_of_birth);
    printf("Total marks: %d\n", record->total_marks);
}

```

```

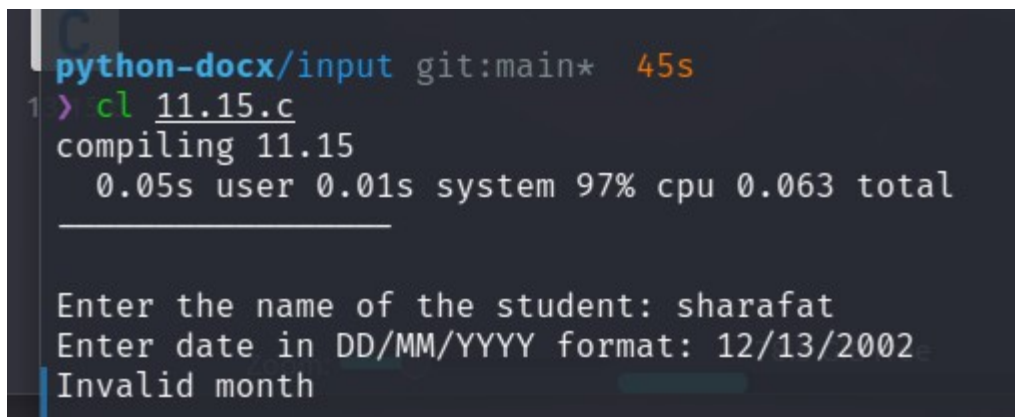
void sort_by_total_marks(struct student_record *record, int number)
{
    int i, j;
    struct student_record temp;
    for (i = 0; i < number; i++)
    {
        for (j = i; j < number; j++)
        {
            if (record[i].total_marks > record[j].total_marks)
            {
                temp = record[i];
                record[i] = record[j];
                record[j] = temp;
            }
        }
    }
}

```

```

int main()
{
    int num = 5;
    struct student_record record[num];
    int i;
    for (i = 0; i < num; i++)
    {
        input_student_record(&record[i]);
    }
    sort_by_total_marks(record, num);
    for (i = 0; i < num; i++)
    {
        print_student_record(&record[i]);
    }
}

```



```

python-docx/input git:main* 45s
1 > cl 11.15.c
compiling 11.15
    0.05s user 0.01s system 97% cpu 0.063 total

Enter the name of the student: sharafat
Enter date in DD/MM/YYYY format: 12/13/2002
Invalid month

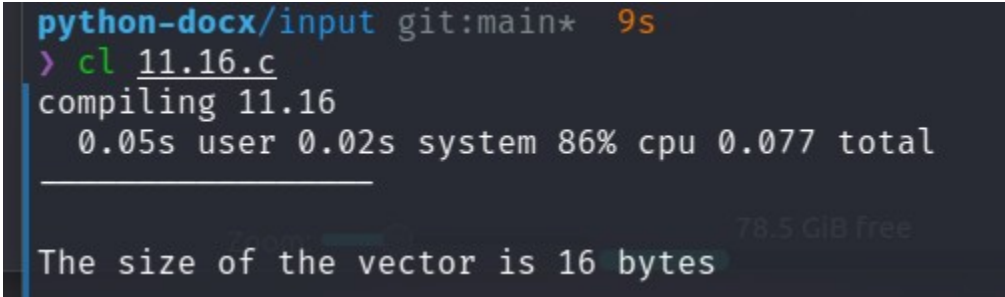
```

### 11.16 Write a C program that prints the size of a structure data type.

```
#include <stdio.h>

struct vector
{
    int *elements;
    int size;
};

int main()
{
    struct vector v;
    printf("The size of the vector is %lu bytes\n", sizeof(v));
    return 0;
}
```



A terminal window with a dark background. The prompt is 'python-docx/input git:main\* 9s'. The user enters 'c\ 11.16.c'. The output shows 'compiling 11.16' followed by timing information: '0.05s user 0.02s system 86% cpu 0.077 total'. A horizontal line is drawn. The output continues with 'The size of the vector is 16 bytes'. In the bottom right corner, the text '78.5 GiB free' is visible.

```
python-docx/input git:main* 9s
> c\ 11.16.c
compiling 11.16
  0.05s user 0.02s system 86% cpu 0.077 total
_____
The size of the vector is 16 bytes
78.5 GiB free
```



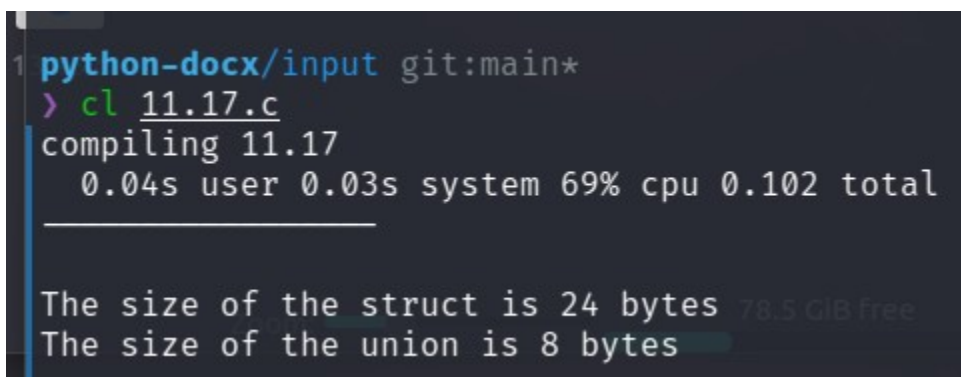
**11.17 Write a C program that prints the size of a structure and union data type that have same number and type of members.**

```
#include <stdio.h>

struct vector
{
    int *elements;
    char size;
    double double_size;
};

union union_vector
{
    int *elements;
    char size;
    double doule_size;
};

int main()
{
    struct vector v;
    union union_vector u;
    printf("The size of the struct is %lu bytes\n", sizeof(v));
    printf("The size of the union is %lu bytes\n", sizeof(u));
    return 0;
}
```



```
1 python-docx/input git:main*
> cl 11.17.c
compiling 11.17
  0.04s user 0.03s system 69% cpu 0.102 total

The size of the struct is 24 bytes
The size of the union is 8 bytes
```

### 11.18 Write a C program for demonstrating operations on individual structure members using pointer notation.

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

struct person
{
    char name[50];
    int age;
    float height;
};

int main()
{
    struct person p1 = {
        .name = "John",
        .age = 25,
        .height = 1.75,
    };
    printf("Original values:\n");
    printf("Name: %s\n", p1.name);
    printf("Age: %d\n", p1.age);
    printf("Height: %.2f\n", p1.height);

    char *name;
    int *age;
    float *height;

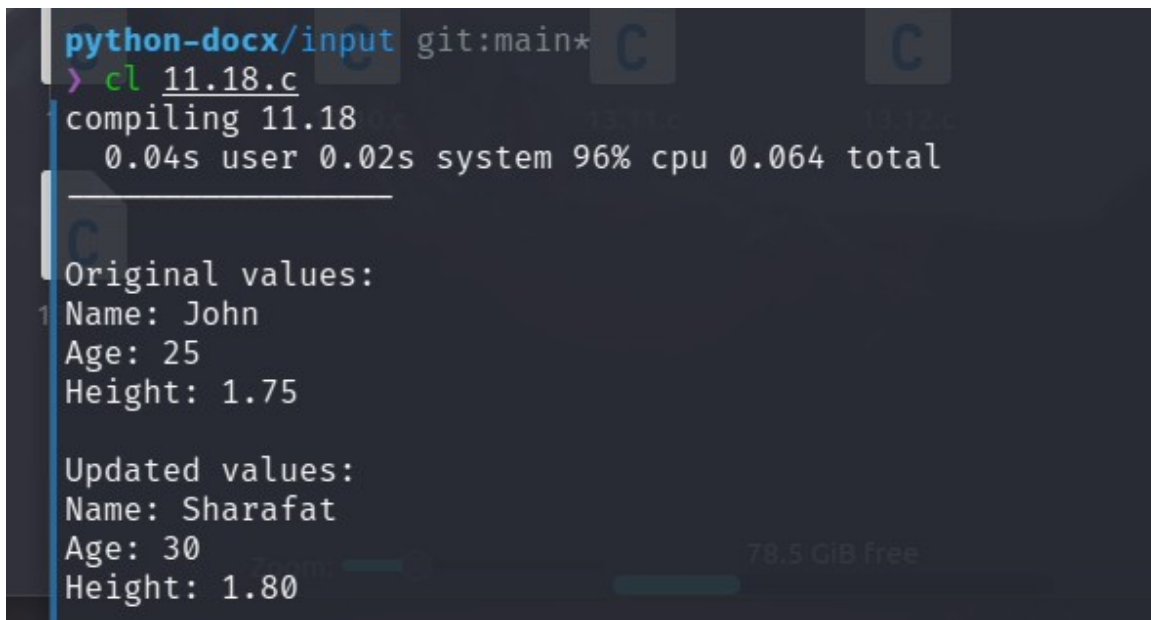
    name = p1.name;
    age = &p1.age;
    height = &p1.height;

    strcpy(name, "Sharafat");
    *age = 30;
```

```
*height = 1.80;

printf("\nUpdated values:\n");
printf("Name: %s\n", p1.name);
printf("Age: %d\n", p1.age);
printf("Height: %.2f\n", p1.height);

return 0;
}
```



```
python-docx/input git:main* C C
> cl 11.18.c
compiling 11.18
0.04s user 0.02s system 96% cpu 0.064 total

Original values:
1 Name: John
Age: 25
Height: 1.75

Updated values:
Name: Sharafat
Age: 30
Height: 1.80
```

## 19 Write a program to copy the contents of one file into another.

```
#include <stdio.h>

int main()
{
    FILE *fp1, *fp2;
    char ch;
    fp1 = fopen("file1.txt", "r");
    fp2 = fopen("file2.txt", "w");
    while ((ch = fgetc(fp1)) != EOF)
    {
        fputc(ch, fp2);
    }
    return 0;
}
```

**20 Two files DATA1 and DATA2 contain sorted lists of integers. Write a program to produce a third file DATA which holds a single sorted, merged list of these two lists. Use command line arguments to specify the file names.**

```
#include <stdio.h>

int main()
{
    FILE *fp1, *fp2, *fp3;
    int num1, num2;
    fp1 = fopen("DATA1", "r");
    fp2 = fopen("DATA2", "r");
    fp3 = fopen("DATA", "w");

    fscanf(fp1, "%d", &num1);
    fscanf(fp2, "%d", &num2);

    while (!feof(fp1) && !feof(fp2))
    {
        if (num1 < num2)
        {
            fprintf(fp3, "%d\n", num1);
            fscanf(fp1, "%d", &num1);
        }
        else
        {
            fprintf(fp3, "%d\n", num2);
            fscanf(fp2, "%d", &num2);
        }
    }
    while (!feof(fp1))
    {
        fprintf(fp3, "%d\n", num1);
        fscanf(fp1, "%d", &num1);
    }
}
```

```
}  
while (!feof(fp2))  
{  
    fprintf(fp3, "%d\n", num2);  
    fscanf(fp2, "%d", &num2);  
}  
return 0;  
}
```

**21 Write a program that compares two files and returns 0 if they are equal and 1 if they are not.**

```
#include <stdio.h>

int main()
{
    FILE *fp1, *fp2;
    int num1, num2;
    fp1 = fopen("DATA1", "r");
    fp2 = fopen("DATA2", "r");

    fscanf(fp1, "%d", &num1);
    fscanf(fp2, "%d", &num2);

    while (!feof(fp1) || !feof(fp2))
    {
        if (num1 != num2)
        {
            printf("Not same");
            return 1;
        }
        fscanf(fp1, "%d", &num1);
        fscanf(fp2, "%d", &num2);
    }
    if (feof(fp1) || feof(fp2))
    {
        printf("Not same");
        return 1;
    }
    printf("Same");
    return 0;
}
```

## 22 Write a program that appends one file at the end of another.

```
#include <stdio.h>

int main()
{
    FILE *fp1, *fp2;
    int num;
    fp1 = fopen("DATA1", "r");
    fp2 = fopen("DATA2", "a");

    fseek(fp2, 0, 2);

    fscanf( fp1, "%d", &num);
    while ( !feof(fp1) )
    {
        fprintf( fp2, "%d\n", num);
        fscanf( fp1, "%d", &num);
    }
    return 0;
}
```



**23 Write a program that reads a file containing integers and appends at its end the sum of all the integers.**

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

```
int main()
```

```
{
```

```
    FILE *fp1, *fp2;
```

```
    int num, sum = 0;
```

```
    fp1 = fopen("DATA", "r");
```

```
    fp2 = fopen("DATA", "a");
```

```
    fscanf(fp1, "%d", &num);
```

```
    while (!feof(fp1))
```

```
    {
```

```
        printf("%d + ", num);
```

```
        sum += num;
```

```
        fscanf(fp1, "%d", &num);
```

```
    }
```

```
    printf("= %d\n", sum);
```

Write a program to copy the contents of one file into another.

```
fseek(fp2, 0, 2);
```

```
fprintf(fp2, "%d\n", sum);
```

```
    return 0;
```

```
}
```

**24 Write a program that prompts the user for two files, one containing a line of text known as source file and other, an empty file known as target file and then copies the contents of source file into target file. Modify the program so that a specified character is deleted from the source file as it is copied to the target file.**

```
#include <stdio.h>

int main()
{
    FILE *fp1, *fp2;
    char ch, exclude;
    fp1 = fopen("file1.txt", "r");
    fp2 = fopen("file2.txt", "w");
    printf("What to exclude? ");
    scanf("%c", &exclude);
    while ((ch = fgetc(fp1)) != EOF)
    {
        if (ch == exclude)
            continue;
        fputc(ch, fp2);
    }
    return 0;
}
```

**25 Write a program that requests for a file name and an integer, known as offset value. The program then reads the file starting from the location specified by the offset value and prints the contents on the screen. Note: If the offset value is a positive integer, then printing skips that many lines. If it is a negative number, it prints that many lines from the end of the file. An appropriate error message should be printed, if anything goes wrong.**

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

int main()
{
    FILE *fp;
    char file_name[100];
    int offset;

    printf("Enter file name: ");
    scanf("%s", file_name);
    printf("Enter offset: ");
    scanf("%d", &offset);

    fp = fopen(file_name, "r");

    if (fp == NULL)
    {
        printf("File not found\n");
        return 1;
    }

    char ch;
    int i = 0;
    if (offset >= 0)
    {
        while ((ch = fgetc(fp)) != EOF)
```

```

{
    if (i >= offset)
    {
        break;
    }
    printf("%c", ch);
    if (ch == '\n')
    {
        i++;
    }
}
}
else
{
    int total_lines = 0;
    while ((ch = fgetc(fp)) != EOF)
    {
        if (ch == '\n')
        {
            total_lines++;
        }
    }
    // printf("Total lines: %d\n", total_lines);
    rewind(fp);
    i = 0;
    while ((ch = fgetc(fp)) != EOF)
    {
        if (total_lines + offset <= i)
        {
            printf("%c", ch);
            // break;
        }
        if (ch == '\n')
        {
            i++;
        }
    }
}

```

```
    }  
    return 0;  
}
```

**26 Write a program to create a sequential file that could store details about five products. Details include product code, cost and number of items available and are provided through keyboard.**

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

```
struct Product
```

```
{  
    int code;  
    float cost;  
    int quantity;  
};
```

```
int main()
```

```
{  
    FILE *fp;  
    fp = fopen("products.txt", "w");  
    struct Product products[5];  
    for (int i = 0; i < 5; i++)  
    {  
        printf("\nEnter product code: ");  
        scanf("%d", &products[i].code);  
        printf("Enter product cost: ");  
        scanf("%f", &products[i].cost);  
        printf("Enter product quantity: ");  
        scanf("%d", &products[i].quantity);  
        fprintf(fp, "%d %f %d\n", products[i].code, products[i].cost,  
products[i].quantity);  
    }  
    fclose(fp);  
    return 0;  
}
```

**27 Write a program to read the file created in Exercise 13.8 and compute and print the total value of all the five products.**

```
#include <stdio.h>

int main()
{
    FILE *fp;
    fp = fopen("products.txt", "r");
    float sum = 0;
    while (feof(fp) == 0)
    {
        float temp;
        fscanf(fp, "%*d %f %*d ", &temp);
        sum += temp;
    }
    fclose(fp);
    printf("Total cost: %f\n", sum);
    return 0;
}
```

**28 Rewrite the program developed in Exercise 13.8 to store the details in a random access file and print the details of alternate products from the file. Modify the program so that it can output the details of a product when its code is specified interactively.**

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

```
struct Product
{
    int code;
    float cost;
    int quantity;
};
```

```
int append_data()
{
    FILE *fp;
    fp = fopen("products.txt", "a");
    struct Product products;
    printf("\nEnter product code: ");
    scanf("%d", &products.code);
    printf("Enter product cost: ");
    scanf("%f", &products.cost);
    printf("Enter product quantity: ");
    scanf("%d", &products.quantity);
    fprintf(fp, "%d %f %d\n", products.code, products.cost, products.quantity);
    fclose(fp);
    return 0;
}
```

```
int filter_with_code(int code)
{
    FILE *fp;
    fp = fopen("products.txt", "r");
    struct Product products[5];
```



```

int i = 0;
while (!feof(fp))
{
    fscanf(fp, "%d %f %d", &products[i].code, &products[i].cost,
&products[i].quantity);
    if (products[i].code == code)
    {
        printf("%d %f %d\n", products[i].code, products[i].cost,
products[i].quantity);
    }
    i++;
}
fclose(fp);
return 0;
}

```

```

int main()
{
    printf("1. Append data\n");
    printf("2. Filter with code\n");
    printf("Enter choice: ");
    int choice;
    scanf("%d", &choice);
    switch (choice)
    {
        case 1:
            append_data();
            break;
        case 2:
            printf("Enter code: ");
            int code;
            scanf("%d", &code);
            filter_with_code(code);
            break;
        default:
            printf("Invalid choice\n");
            break;
    }
}

```

```
    }  
    return 0;  
}
```

**29 Write a C program that uses file handling methods to store records of mixed data in a file.**

```
#include <stdio.h>

struct Product
{
    int code;
    char name[20];
    float cost;
    int quantity;
};

int main()
{
    FILE *fp;
    fp = fopen("products.txt", "w");
    struct Product products[5];
    for (int i = 0; i < 5; i++)
    {
        printf("\nEnter product code: ");
        scanf("%d", &products[i].code);
        printf("Enter product name: ");
        scanf("%s", products[i].name);
        printf("Enter product cost: ");
        scanf("%f", &products[i].cost);
        printf("Enter product quantity: ");
        scanf("%d", &products[i].quantity);
        fprintf(fp, "%d %f %d\n", products[i].code, products[i].cost,
products[i].quantity);
    }
    fclose(fp);
    return 0;
}
```



**30 Write a C program that uses getw function to read integer values from one file. Subsequently, it uses the putw function to write the integer values in reverse order in another file.**

// This program may not work perfectly in some systems,  
// so consider avoiding getw, putw, etc. in your programs.

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

```
int main()
```

```
{
```

```
    FILE *fp1, *fp2;
```

```
    fp1 = fopen("file1.txt", "r");
```

```
    fp2 = fopen("file2.txt", "w");
```

```
    int num, count = 0;
```

```
    while (feof(fp1) == 0)
```

```
    {
```

```
        fscanf(fp1, "%d ", &num);
```

```
        count++;
```

```
    }
```

```
    printf("Number of elements in file1.txt: %d\n", count);
```

```
    rewind(fp1);
```

```
    // char arr[count];
```

```
    for (int i = 0; i < count; i++)
```

```
    {
```

```
        // fscanf(fp1, "%d", &arr[i]);
```

```
        // arr[i] = getw(fp1);
```

```
        putw(getw(fp1), fp2);
```

```
    }
```

```
    // for (int i = 0; i < count; i++)
```

```
    // {
```

```
        // putw(arr[i], fp2);
```

```
        // fprintf(fp2, "%d ", arr[i]);  
    // }  
  
    fclose(fp1);  
    fclose(fp2);  
  
    printf("--- END ---\n");  
    return 0;  
}
```

**31 Write a C program that reads characters from a file and prints their ASCII codes.**

```
#include <stdio.h>
int main()
{
    FILE *fp;
    fp = fopen("DATA", "r");
    char ch;
    while (feof(fp) == 0)
    {
        ch =getc(fp);
        printf("%c - ASCII VALUE = %d\n", ch, (int)ch);
    }
    return 0;
}
```

**32 Write a C program that concatenates the contents of two files and writes then in the third file.**

```
#include <stdio.h>

int main()
{
    FILE *fp1, *fp2, *fp3;
    int num;
    fp1 = fopen("DATA1", "r");
    fp2 = fopen("DATA2", "r");
    fp3 = fopen("DATA", "w");

    fscanf( fp1, "%d", &num);
    while ( !feof(fp1) )
    {
        fprintf( fp3, "%d\n", num);
        fscanf( fp1, "%d", &num);
    }

    fscanf( fp2, "%d", &num);
    while ( !feof(fp2) )
    {
        fprintf( fp3, "%d\n", num);
        fscanf( fp2, "%d", &num);
    }
    return 0;
}
```



**33 Write a C program that uses fscanf function to read integer values from a file, computes the square of each integer value and places the resultant values in a different file.**

```
#include <stdio.h>

int main()
{
    FILE *fp1, *fp2;
    int num;
    fp1 = fopen("DATA1", "r");
    fp2 = fopen("DATA2", "w");

    fseek(fp2, 0, 2);

    fscanf( fp1, "%d", &num);
    while ( !feof(fp1) )
    {
        fprintf( fp2, "%d\n", num * num);
        fscanf( fp1, "%d", &num);
    }
    return 0;
}
```