**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](#__RefHeading___Toc3091_3274203059)

[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](#__RefHeading___Toc3093_3274203059)

[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](#__RefHeading___Toc3095_3274203059)

[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](#__RefHeading___Toc3097_3274203059)

[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](#__RefHeading___Toc3099_3274203059)

[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](#__RefHeading___Toc3101_3274203059)

[11.7 Add a function called nextdate to the program designed in Exercise 11.4 to perform the following task: 18](#__RefHeading___Toc3103_3274203059)

[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](#__RefHeading___Toc3105_3274203059)

[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](#__RefHeading___Toc3107_3274203059)

[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](#__RefHeading___Toc3109_3274203059)

[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](#__RefHeading___Toc3111_3274203059)

[11.12 Define a structure named census with the following three members: 34](#__RefHeading___Toc3113_3274203059)

[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](#__RefHeading___Toc3115_3274203059)

[11.14 Define a structure called cricket that will describe the following information: 42](#__RefHeading___Toc3117_3274203059)

[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](#__RefHeading___Toc3119_3274203059)

[11.16 Write a C program that prints the size of a structure data type. 48](#__RefHeading___Toc3121_3274203059)

[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](#__RefHeading___Toc3123_3274203059)

[11.18 Write a C program for demonstrating operations on individual structure members using pointer notation. 50](#__RefHeading___Toc3125_3274203059)

[19 Write a program to copy the contents of one file into another. 52](#__RefHeading___Toc3127_3274203059)

[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](#__RefHeading___Toc3129_3274203059)

[21 Write a program that compares two files and returns 0 if they are equal and 1 is they are not. 55](#__RefHeading___Toc3131_3274203059)

[22 Write a program that appends one file at the end of another. 56](#__RefHeading___Toc3133_3274203059)

[23 Write a program that reads a file containing integers and appends at its end the sum of all the integers. 57](#__RefHeading___Toc3135_3274203059)

[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](#__RefHeading___Toc3137_3274203059)

[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](#__RefHeading___Toc3139_3274203059)

[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](#__RefHeading___Toc3141_3274203059)

[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](#__RefHeading___Toc3143_3274203059)

[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](#__RefHeading___Toc3145_3274203059)

[29 Write a C program that uses file handling methods to store records of mixed data in a file. 67](#__RefHeading___Toc3147_3274203059)

[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](#__RefHeading___Toc3149_3274203059)

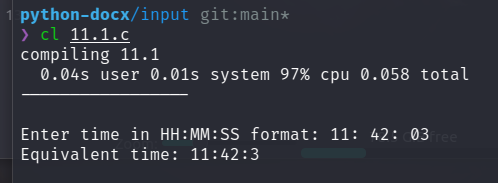
[31 Write a C program that reads characters from a file and prints their ASCII codes. 71](#__RefHeading___Toc3151_3274203059)

[32 Write a C program that concatenates the contents of two files and writes then in the third file. 72](#__RefHeading___Toc3153_3274203059)

[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](#__RefHeading___Toc3155_3274203059)

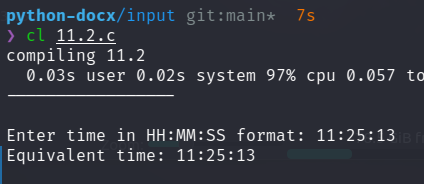
# 11.1 Define a structure data type called time\_struct containing three members integer hour, integer minute and integer second . Develop a program that would assign values to the individual members and display the time in the following form:

#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;  
}



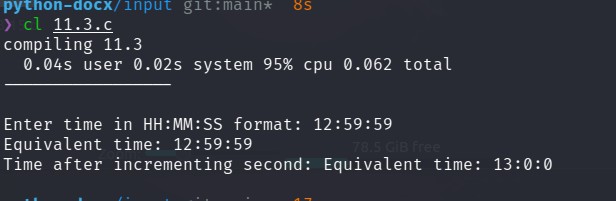
# 11.2 Modify the above program such that a function is used to input values to the members and another function to display the time.

#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;  
}



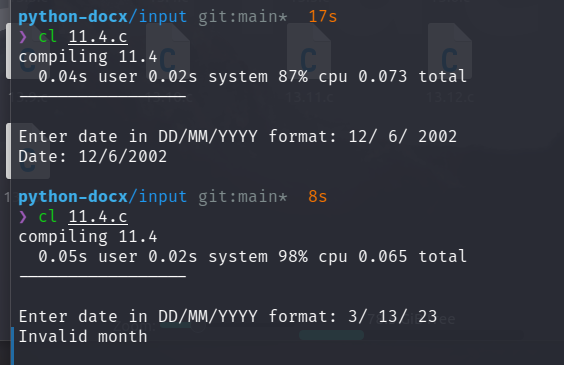
# 11.3 Design a function update that would accept the data structure designed in Exercise 11.1 and increments time by one second and returns the new time. (If the increment results in 60 seconds, then the second member is set to zero and the minute member is incremented by one. Then, if the result is 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;  
}



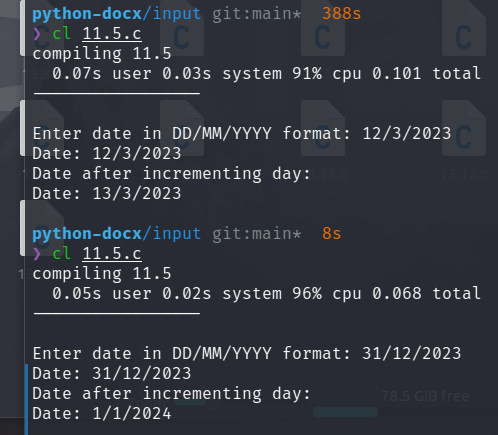
# 11.4 Define a structure data type named date containing three integer members day , month , and year. Develop an interactive modular program to perform the following tasks:

#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;  
}



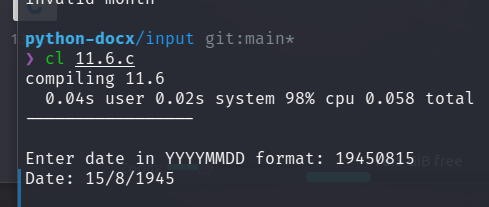
# 11.5 Design a function update that accepts the date structure designed in Exercise 11.4 to increment the date by one day and return the new date. The following rules are applicable:

#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;  
}



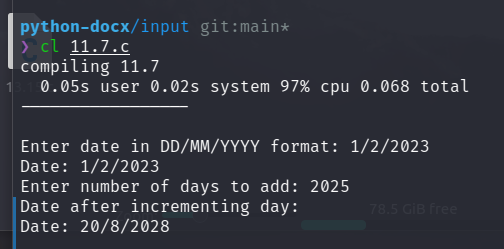
# 11.6 Modify the input function used in Exercise 11.4 such that it reads a value that represents the date in the form of a long integer, like 19450815 for the date 15-8-1945 (August 15, 1945) and assigns suitable values to the members day, month , and year.

#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;  
}



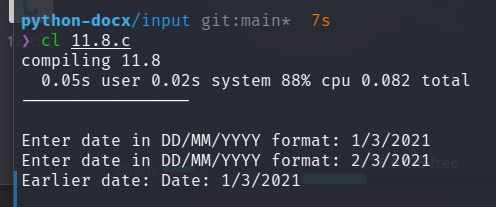
# 11.7 Add a function called nextdate to the program designed in Exercise 11.4 to perform the following task:

#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;  
}



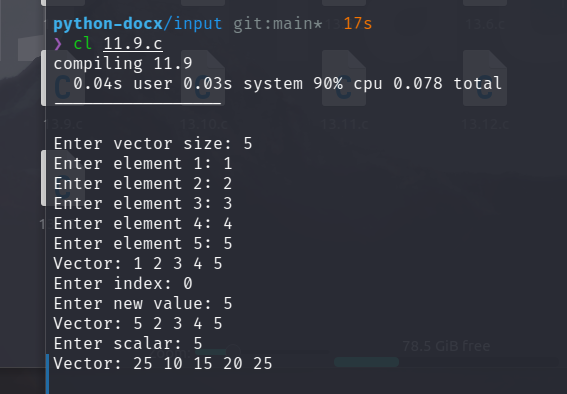
# 11.8 Use the date structure defined in Exercise 11.4 to store two dates. Develop a function that will take these two dates as input and compares them.

#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;  
}



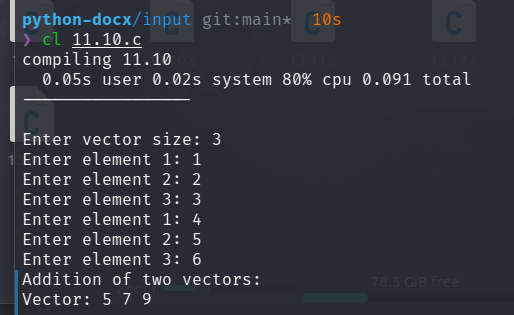
# 11.9 Define a structure to represent a vector (a series of integer values) and write a modular program to perform the following tasks:

#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);  
}



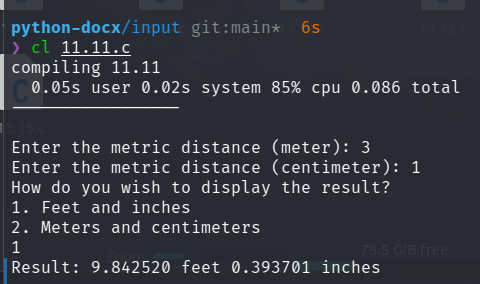
# 11.10 Add a function to the program of Exercise 11.9 that accepts two vectors as input parameters and return the addition of two vectors.

#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);  
}



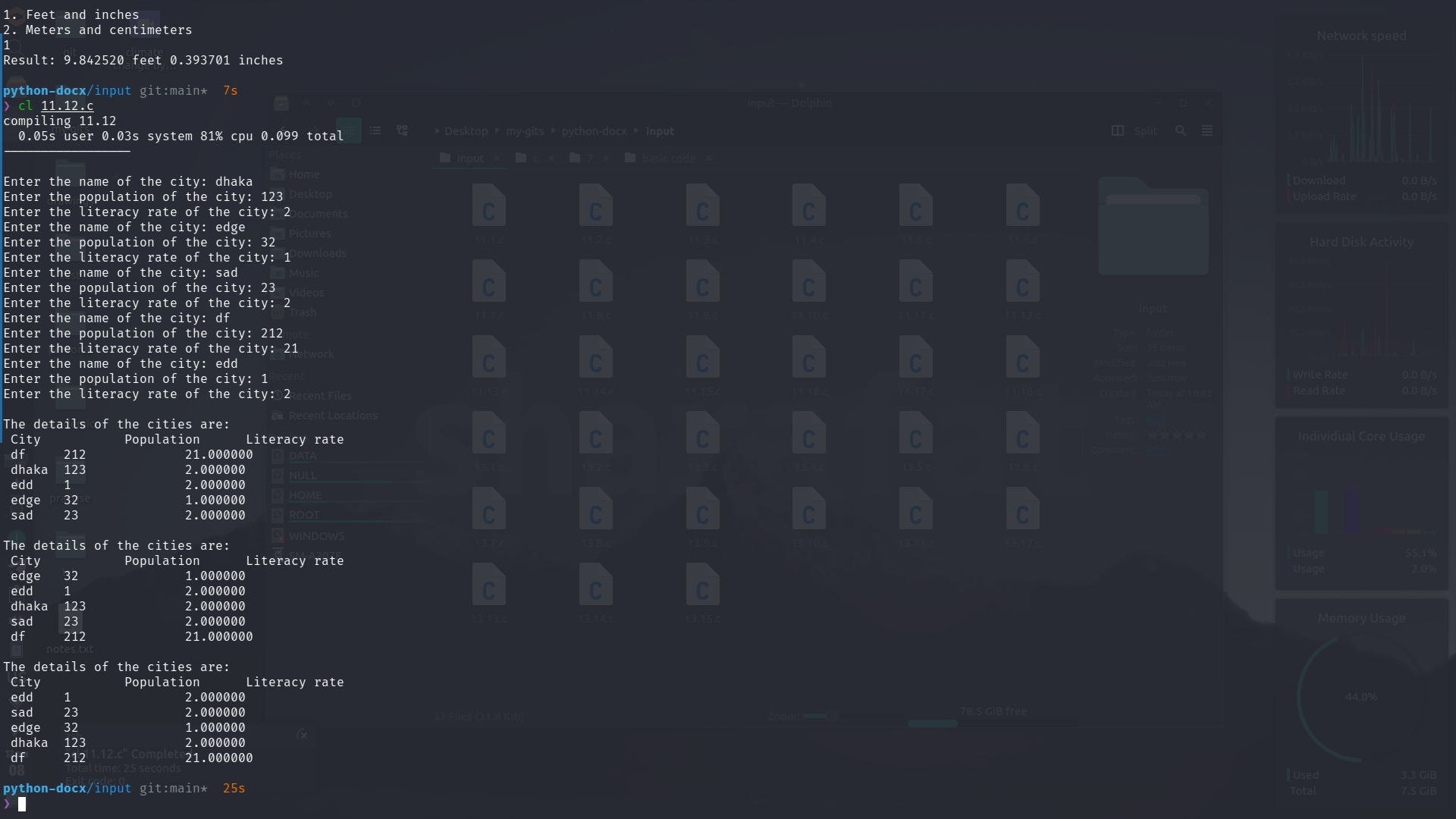
# 11.11 Create two structures named metric and British which store the values of distances. The metric structure stores the values in metres and centimetres and the British structure stores the values in feet and inches. Write a program that reads values for the structure variables and adds values contained 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;   
}



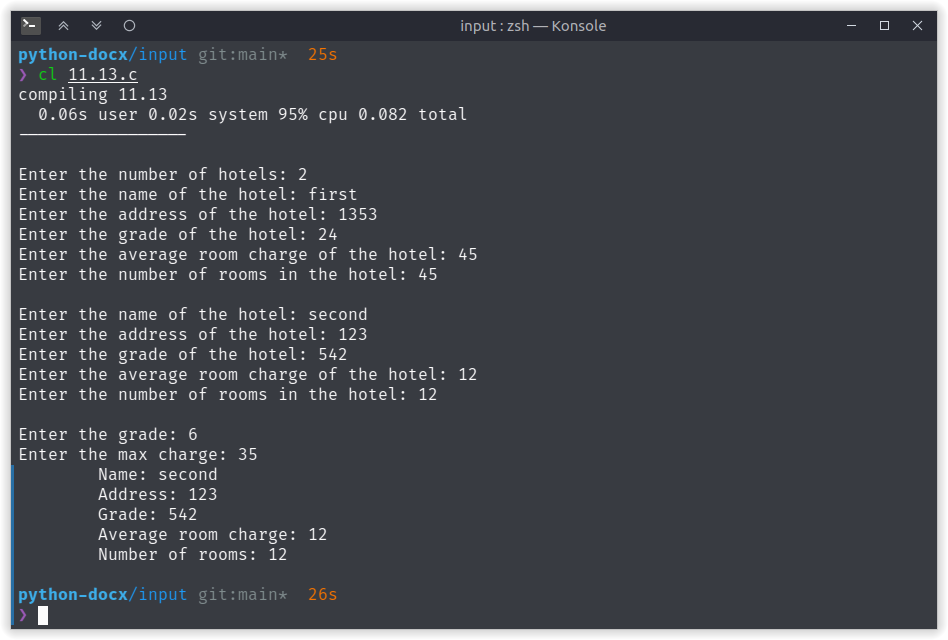
# 11.12 Define a structure named census with the following three members:

#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%ld\t\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;  
}



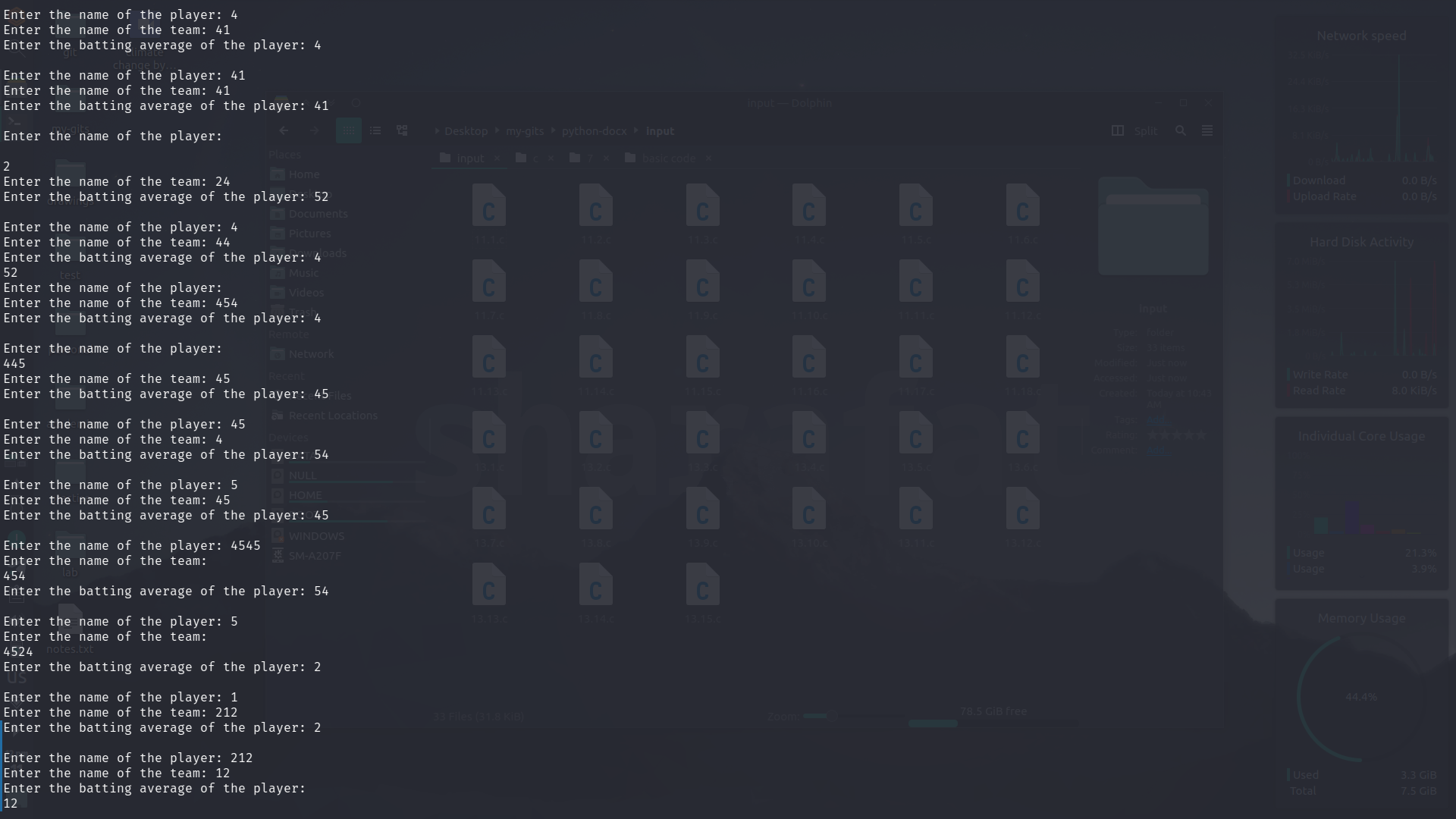
# 11.13 Define a structure that can describe an hotel. It should have members that include the name, address, grade, average room charge, and number of rooms. Write functions to perform the following operations:

#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);  
}



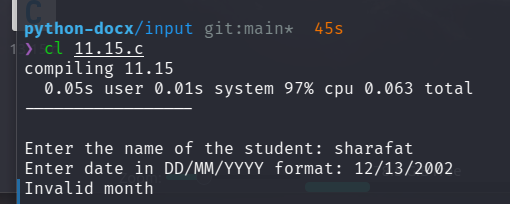
# 11.14 Define a structure called cricket that will describe the following information:

#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;  
}



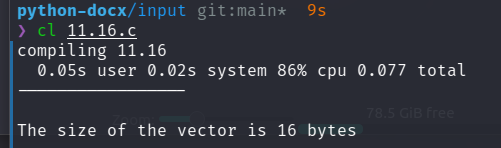
# 11.15 Design a structure student\_record to contain name, date of birth, and total marks obtained. Use the date structure designed in Exercise 11.4 to represent the date of birth. Develop a program to read data for 10 students in a class and list them rank-wise.

#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]);  
 }  
}



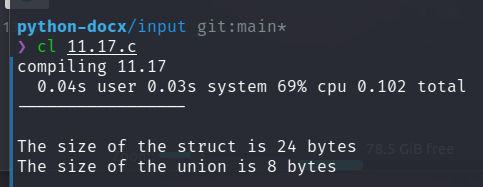
# 11.16 Write a C program that prints the size of a structure data type.

#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;  
}



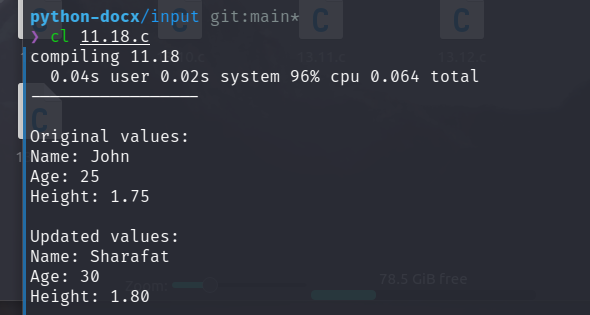
# 11.17 Write a C program that prints the size of a structure and union data type that have same number and type of members.

#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;  
}



# 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;   
}



# 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 is 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;  
}