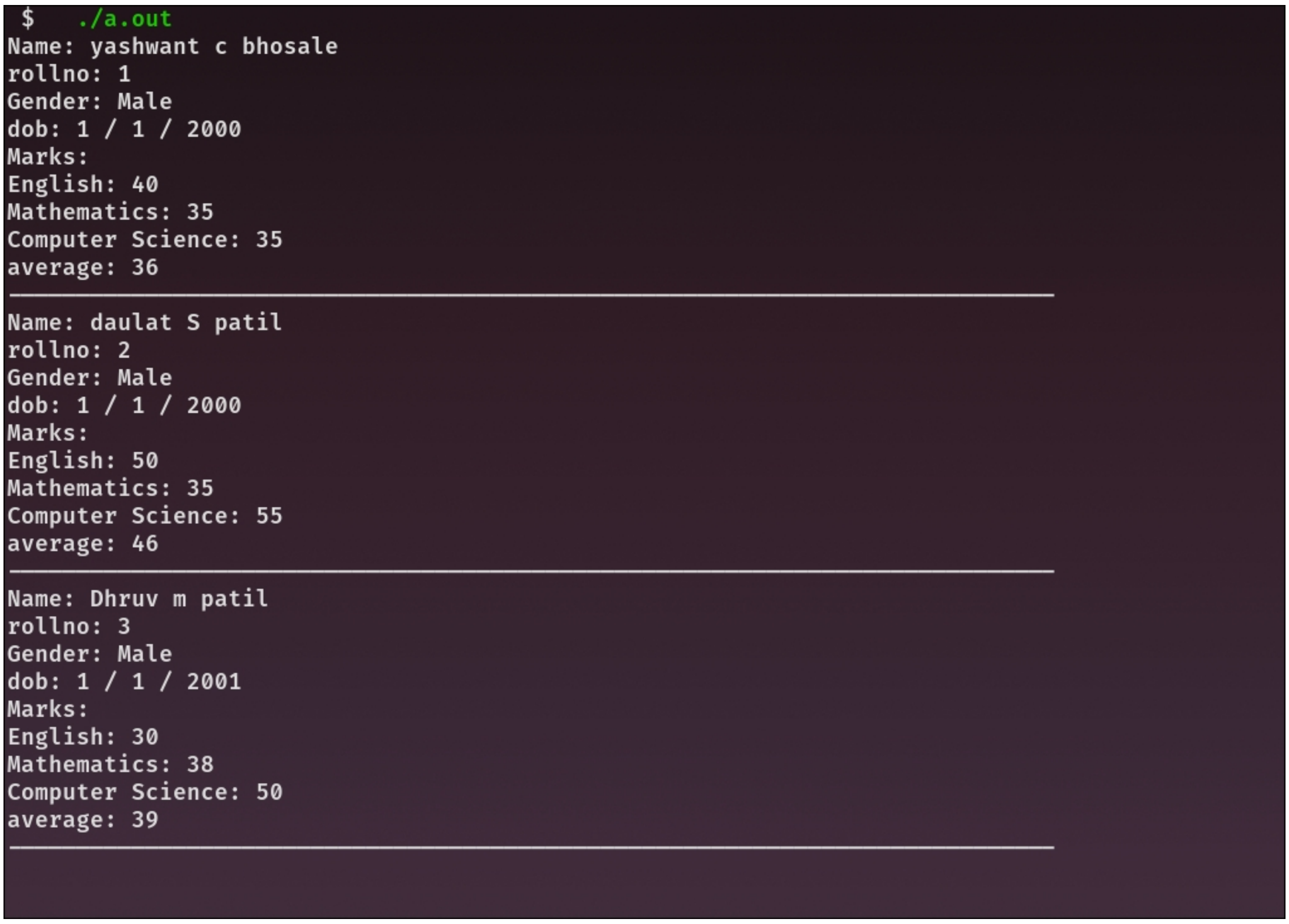
Name: Yashwant C. Bhosale
MIS: 612303039
Div: SY comp div 1

Q1. Q.1 Declare a structure that represents the following hierarchical information: to Remember (a) Student (b) Roll Number …
code:

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

typedef struct {
    char firstName[64];
    char middleName[64];
    char lastName[64];
} Name;

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

typedef struct  {
    int english_marks;
    int mathematics_marks;
    int cs_marks;
} marks;

typedef struct data {
    Name name;
    int rollno;
    char gender[64];
    date date;
    marks marks;
} student_data;

student_data *students[512];
int data_pointer = 0;

void init() {
    for(int i = 0; i < 512; i++){
        students[i] = NULL;
    }
}

void append_student(Name name, int rollno, char gender[], date date,
marks marks) {
    if (data_pointer >= 512) {
        printf("Error: Cannot add more students, array is full.\n");
        return;
    }
    students[data_pointer] = (student_data
*)malloc(sizeof(student_data));

    strcpy(students[data_pointer]->name.firstName, name.firstName);
    strcpy(students[data_pointer]->name.middleName, name.middleName);
    strcpy(students[data_pointer]->name.lastName, name.lastName);

    students[data_pointer]->rollno = rollno;

    strcpy(students[data_pointer]->gender, gender);

    students[data_pointer]->date.day = date.day;
    students[data_pointer]->date.month = date.month;
    students[data_pointer]->date.year = date.year;

    students[data_pointer]->marks.english_marks = marks.english_marks;
    students[data_pointer]->marks.mathematics_marks =
marks.mathematics_marks;
    students[data_pointer]->marks.cs_marks = marks.cs_marks;

    data_pointer++;
}

void display_students() {
    int i = 0;
    while(students[i] != NULL) {
        printf("Name: %s ", students[i]->name.firstName);
        printf("%s ", students[i]->name.middleName);
        printf("%s\n", students[i]->name.lastName);
        printf("rollno: %d\n", students[i]->rollno);
        printf("Gender: %s\n", students[i]->gender);
        printf("dob: %d / %d / %d\n", students[i]->date.day,
students[i]->date.month, students[i]->date.year);
        printf("Marks:\nEnglish: %d\nMathematics: %d\nComputer Science:
%d\n", students[i]->marks.english_marks, students[i]-
>marks.mathematics_marks, students[i]->marks.cs_marks);
        printf("average: %d\n", (students[i]->marks.english_marks +
students[i]->marks.mathematics_marks+students[i]->marks.cs_marks)/3);

printf("-------------------------------------------------------------
------------\n");
        i++;
    }
}
```

```
void display_low_aggreagate_students() {
    int i = 0;
    while(students[i] != NULL) {
                 if(((students[i]->marks.english_marks  +  students[i]-
>marks.mathematics_marks+students[i]->marks.cs_marks))/3 < 40){
        printf("Name: %s ", students[i]->name.firstName);
        printf("%s ", students[i]->name.middleName);
        printf("%s\n", students[i]->name.lastName);
           printf("average: %d\n", (students[i]->marks.english_marks +
students[i]->marks.mathematics_marks+students[i]->marks.cs_marks)/3);
        printf("----------------------------------------------------
--------------------\n");
        }
        i++;
    }
}

int main() {
    init();
    char gender[] = "Male";
    int rollno = 1;
    marks student_marks1 = {40, 35, 35};
    // marks student_marks2 = {60, 60, 60};
    marks student_marks3 = {50, 35, 55};
    marks student_marks4 = {30, 38, 50};
    // marks student_marks5 = {55, 50, 45};
    date birthdate = {1, 1, 2000};
    date birthdate2 = {1, 1, 2001};

    Name name1 = {"yashwant", "c", "bhosale"};
    append_student(name1, rollno++, gender, birthdate, student_marks1);

    // Name name2 = {"manohar", "a", "jadhav"};
    // append_student(name2, rollno++, gender, birthdate,
student_marks2);

    Name name3 = {"daulat", "S", "patil"};
    append_student(name3, rollno++, gender, birthdate, student_marks3);

    Name name4 = {"Dhruv", "m", "patil"};
    append_student(name4, rollno++, gender, birthdate2,
student_marks4);

    // Name name5 = {"Rishabh", "R", "sharma"};
    // append_student(name5, rollno++, gender, birthdate2,
student_marks5);

    display_students();
    printf("\n\n\n");
    printf("Low aggregate students\n");
    display_low_aggreagate_students();
}
```

Output:

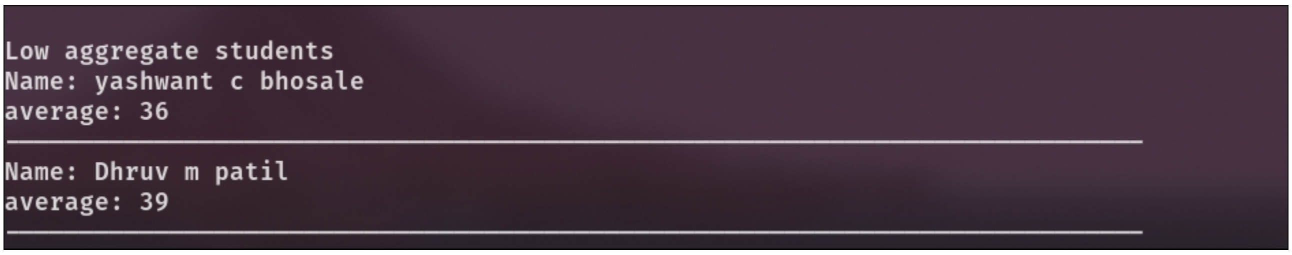


```
 $   ./a.out
Name: yashwant c bhosale
rollno: 1
Gender: Male
dob: 1 / 1 / 2000
Marks:
English: 40
Mathematics: 35
Computer Science: 35
average: 36
────────────────────────────────────────────────────────────────
Name: daulat S patil
rollno: 2
Gender: Male
dob: 1 / 1 / 2000
Marks:
English: 50
Mathematics: 35
Computer Science: 55
average: 46
────────────────────────────────────────────────────────────────
Name: Dhruv m patil
rollno: 3
Gender: Male
dob: 1 / 1 / 2001
Marks:
English: 30
Mathematics: 38
Computer Science: 50
average: 39
────────────────────────────────────────────────────────────────
```

**Q2. Q.2 Using the above structure, write a program to display the details of the student whose
name is entered by the user. Display the name of the students who have secured less than 40% of the aggregate. In addition, print each student's average marks.**

Code:

```
// void display_low_aggreagate_students() in above Q1 code
```

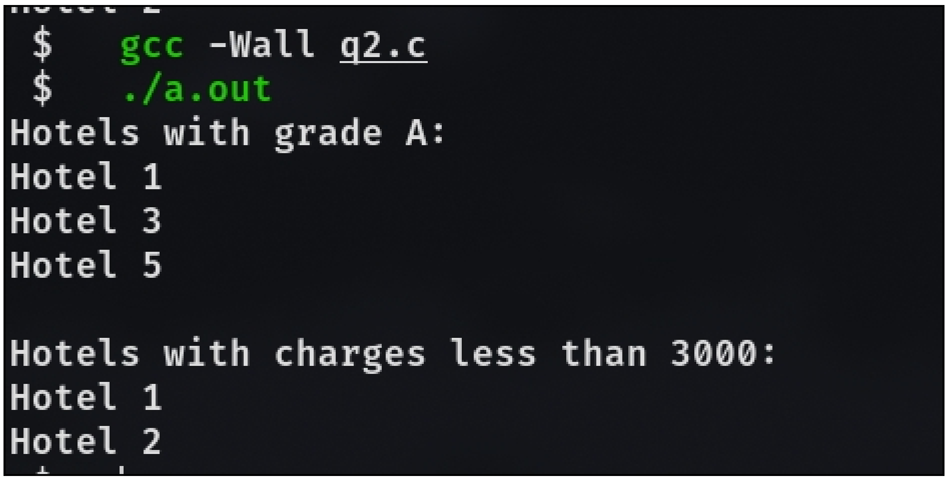


```
Low aggregate students
Name: yashwant c bhosale
average: 36
────────────────────────────────────────────────────────────────
Name: Dhruv m patil
average: 39
────────────────────────────────────────────────────────────────
```

**Q2. Q.3 Write a program to define a structure for a hotel that has members– name, address,
grade, number of rooms, and room charges. Write a function to print the names of hotels in
a particular grade. Also write a function to print names of hotels that have room charges less
than the specified value**

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

typedef struct {
    char name[128];
    char address[128];
    char grade;
    int no_of_rooms;
    int charges;
} hotel;

hotel *hotels[64];
int hotel_pointer = 0;


void init() {
    for(int i = 0; i < 64; i ++) {
        hotels[i] = NULL;
    }
    return;
}

void append_member(char name[], char address[], char grade, int
no_of_rooms, int charges) {
    if (hotel_pointer >= 64) {
        return;
    }
    hotels[hotel_pointer] = (hotel *)malloc(sizeof(hotel));
    strcpy(hotels[hotel_pointer]->name, name);
    strcpy(hotels[hotel_pointer]->address, address);
    hotels[hotel_pointer]->grade = grade;
    hotels[hotel_pointer]->no_of_rooms = no_of_rooms;
    hotels[hotel_pointer]->charges = charges;
    hotel_pointer++;
    return;
}

void filter_by_grade(char grade) {
    for(int i = 0; i < 64; i++) {
        if (hotels[i] != NULL && hotels[i]->grade == grade) {
            printf("%s\n", hotels[i]->name);
        }
    }
    return;
}
```

```
void filter_by_charges(int charges) {
    for(int i = 0; i < 64; i++) {
        if (hotels[i] != NULL && hotels[i]->charges < charges) {
            printf("%s\n", hotels[i]->name);
        }
    }
    return;
}


int main() {
    init();
    append_member("Hotel 1", "Address 1", 'A', 100, 1000);
    append_member("Hotel 2", "Address 2", 'B', 200, 2000);
    append_member("Hotel 3", "Address 3", 'A', 300, 3000);
    append_member("Hotel 4", "Address 4", 'B', 400, 4000);
    append_member("Hotel 5", "Address 5", 'A', 500, 5000);
    append_member("Hotel 6", "Address 6", 'B', 600, 6000);
    printf("Hotels with grade A:\n");
    filter_by_grade('A');
    printf("\n");
    printf("Hotels with charges less than 3000:\n");
    filter_by_charges(3000);
    return 0;
}
```
Output:

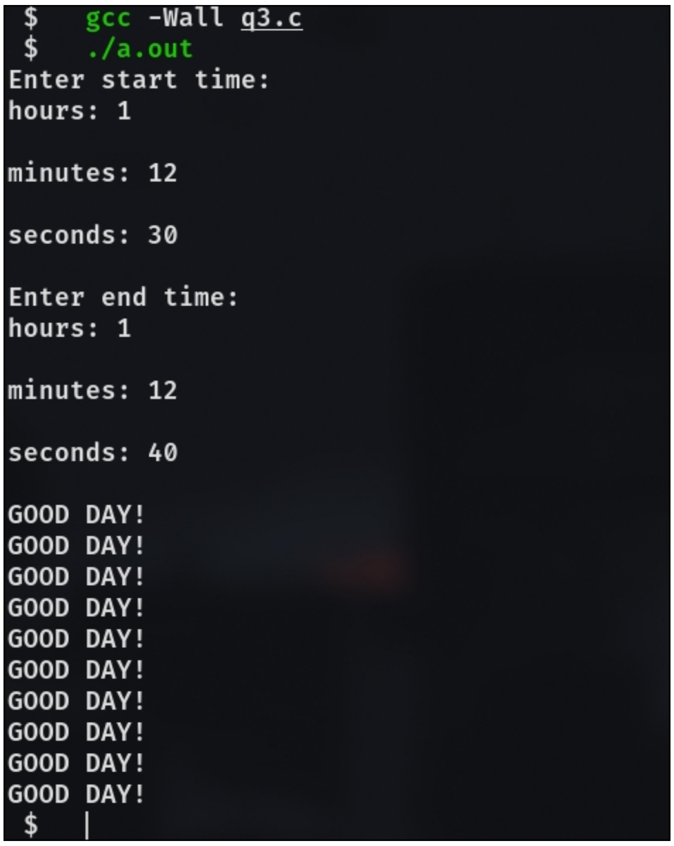


```
 $   gcc -Wall q2.c
 $   ./a.out
Hotels with grade A:
Hotel 1
Hotel 3
Hotel 5

Hotels with charges less than 3000:
Hotel 1
Hotel 2
```

Q3. Declare a structure time that has three fields—hr, min, sec. Create two variables start_time and end_time. Input their values from the user. Then while start_time does not reach the end_time, display GOOD DAY on the screen.

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

typedef struct time {
     unsigned int hours;
     unsigned int minutes;
     unsigned int seconds;
}time;

void print(time start_time, time end_time) {
     printf("\n");
     while((start_time.hours != end_time.hours) ||
(start_time.minutes != end_time.minutes) || (start_time.seconds !=
end_time.seconds)) {
          start_time.seconds++;
          if(start_time.seconds == 60)
               start_time.minutes++;
          if(start_time.minutes == 60)
               start_time.hours++;
          printf("GOOD DAY!\n");
     }
}

int main() {
     time start_time, end_time;
     printf("Enter start time: \n");
     printf("hours: ");
     scanf("%d", &start_time.hours);
     printf("\nminutes: ");
     scanf("%d", &start_time.minutes);
     printf("\nseconds: ");
     scanf("%d", &start_time.seconds);

     printf("\n");

     printf("Enter end time: \n");
     printf("hours: ");
     scanf("%d", &end_time.hours);
     printf("\nminutes: ");
     scanf("%d", &end_time.minutes);
     printf("\nseconds: ");
     scanf("%d", &end_time.seconds);

     print(start_time, end_time);
     return 0;
}
```

Output:

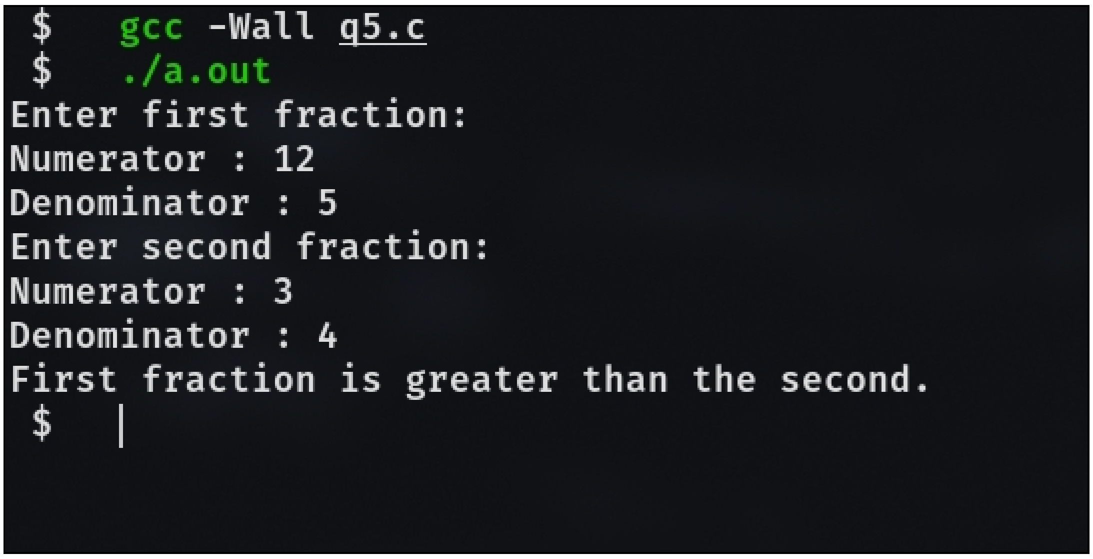


```
 $   gcc -Wall q3.c
 $   ./a.out
Enter start time:
hours: 1

minutes: 12

seconds: 30

Enter end time:
hours: 1

minutes: 12

seconds: 40

GOOD DAY!
GOOD DAY!
GOOD DAY!
GOOD DAY!
GOOD DAY!
GOOD DAY!
GOOD DAY!
GOOD DAY!
GOOD DAY!
GOOD DAY!
 $   |
```

Q.5 Declare a structure fraction that has two fields– numerator and denominator. Create two variables and compare them using function. Return 0 if the two fractions are equal, -1 if the first fraction is less than the second and 1 otherwise. You may convert a fraction into a floating point number for your convenience.
Code:

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

typedef struct fraction {
	float numerator;
	float denominator;
}fraction;

int compare(fraction f1, fraction f2) {
	float fraction_1 = f1.numerator / f1.denominator;
	float fraction_2 = f2.numerator / f2.denominator;

	if(fraction_1 > fraction_2)
		return 1;
	else if(fraction_1 < fraction_2)
		return 0;
	else
		return -1;

}
```

```
int main() {
     fraction f1, f2;
     printf("Enter first fraction: \n");
     printf("Numerator : ");
     scanf("%f", &f1.numerator);
     printf("Denominator : ");
     scanf("%f", &f1.denominator);
     while(f1.denominator == 0) {
          printf("invalid denominator!\nDenominator: ");
          scanf("%f", &f1.denominator);
     }

     printf("Enter second fraction: \n");
     printf("Numerator : ");
     scanf("%f", &f2.numerator);
     printf("Denominator : ");
     scanf("%f", &f2.denominator);

     while(f2.denominator == 0) {
          printf("invalid denominator!\nDenominator: ");
          scanf("%f", &f2.denominator);
     }

     int result  = compare(f1, f2);

     if(!result)
          printf("Second fraction is greater than the first.\n");
     else if(result == 1)
          printf("First fraction is greater than the second.\n");
     else
          printf("Both fractions are equal.\n");

     return 0;
}
```
output:



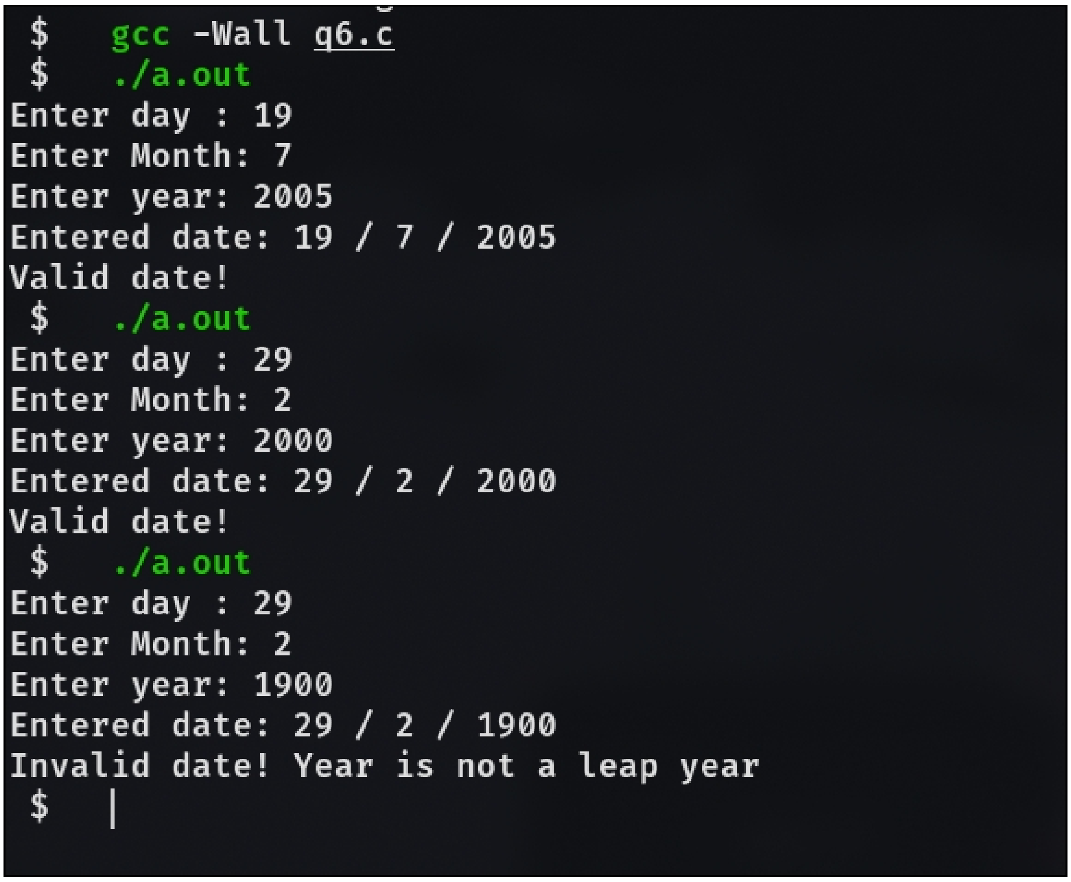
```
 $   gcc -Wall q5.c
 $   ./a.out
Enter first fraction:
Numerator : 12
Denominator : 5
Enter second fraction:
Numerator : 3
Denominator : 4
First fraction is greater than the second.
 $   |
```

Q.6 Define a structure date containing three integers— day, month, and year. Write a program using functions to read data, to validate the date entered by the user and then print the date on the screen. For example, if you enter 29/2/2010 then that is an invalid date as 2010 is not a leap year. Similarly 31/6/2007 is invalid as June does not have 31 days

code:

```
#include <stdio.h>

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

void read_date(date* d) {
    printf("Enter day : ");
    scanf("%d", &d->day);
    printf("Enter Month: ");
    scanf("%d", &d->month);
    printf("Enter year: ");
    scanf("%d", &d->year);
    return;
}

void print_date(date d) {
    printf("Entered date: ");
    printf("%d / %d / %d\n", d.day, d.month, d.year);
    return;
}

void validate_date(date d) {
    if(d.day > 31 || d.day < 0) {
        printf("Invalid date! Day should be less than 31.\n");
        return;
    }
    if(d.month > 12 || d.month < 0){
        printf("Invalid Month!\n");
        return;
    }
    if(d.year < 0) {
        printf("Invalid year!\n");
        return;
    }
    if(d.day == 29 && d.month == 2) {
        if(d.year%4 != 0) {
            printf("Invalid date! Year is not a leap year\n");
            return;
        }
        if(d.year%100 == 0 && d.year%400 != 0) {
            printf("Invalid date! Year is not a leap year\n");
            return;
        }
    }
    if(d.day >= 30 && d.month == 2){
     printf("February cannot have more than 29 days!\n");
     return;
    }
    if((d.day == 31) && (d.month == 4 || d.month == 6 || d.month == 9
|| d.month == 11)){
        printf("Invalid date! No more than 30 days for this month.\n");
        return;
    }
    printf("Valid date!\n");
    return;
}

int main () {
    date new_date;
    read_date(&new_date);
    print_date(new_date);
    validate_date(new_date);
    return 0;
}
```

Output:



```
 $   gcc -Wall q6.c
 $   ./a.out
Enter day : 19
Enter Month: 7
Enter year: 2005
Entered date: 19 / 7 / 2005
Valid date!
 $   ./a.out
Enter day : 29
Enter Month: 2
Enter year: 2000
Entered date: 29 / 2 / 2000
Valid date!
 $   ./a.out
Enter day : 29
Enter Month: 2
Enter year: 1900
Entered date: 29 / 2 / 1900
Invalid date! Year is not a leap year
 $   |
```

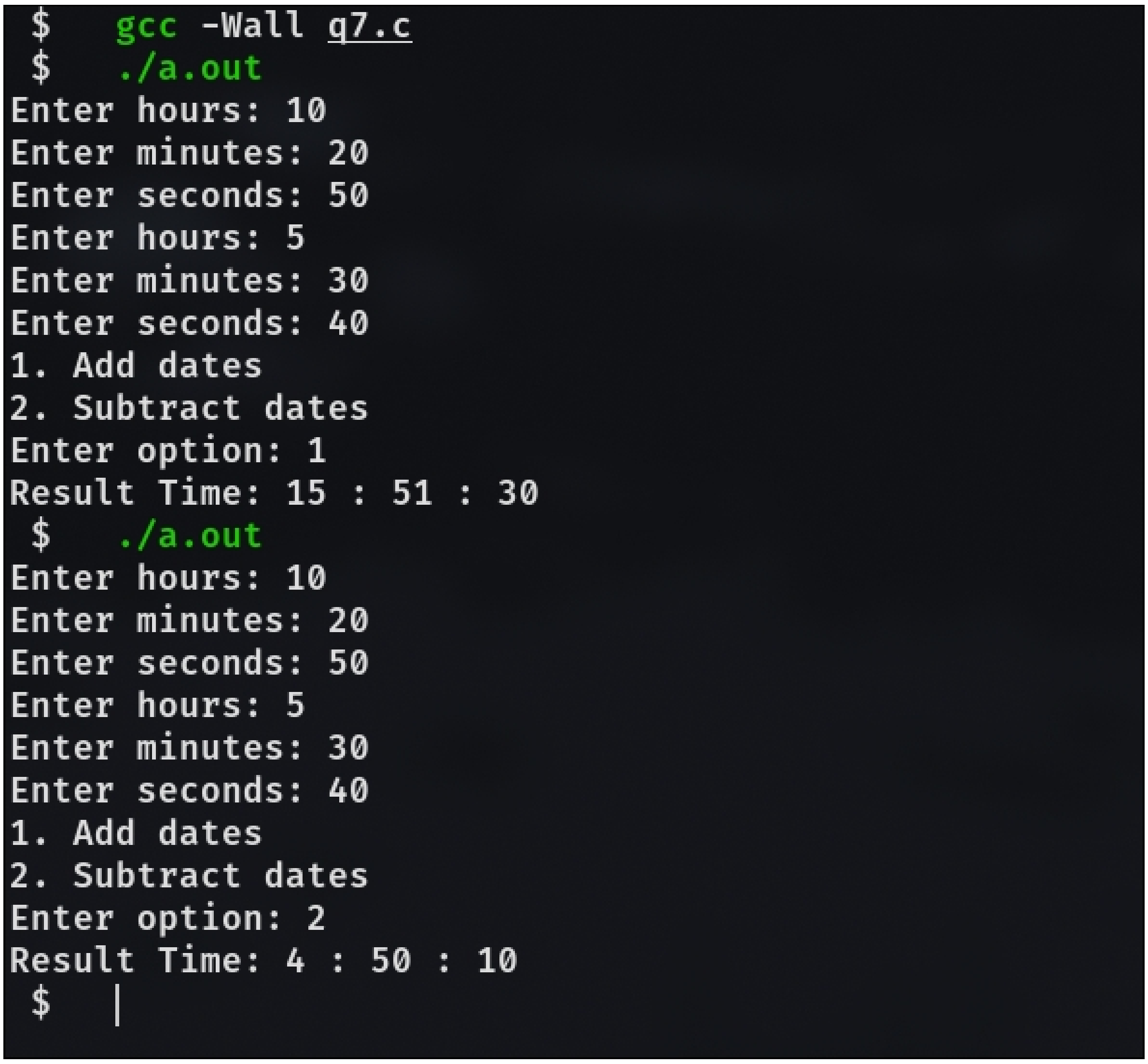
Q.7 Write a program to add and subtract 10hrs 20mins 50sec and 5hrs 30min 40sec.
Code:

// Code is generalized for any time input given

```
#include <stdio.h>
#include <stdlib.h>
typedef struct {
    int hours;
    int minutes;
    int seconds;
} time;

void read_time(time *t) {
    printf("Enter hours: ");
    scanf("%d", &t->hours);
    printf("Enter minutes: ");
    scanf("%d", &t->minutes);
    printf("Enter seconds: ");
    scanf("%d", &t->seconds);
    return;
}

void print_time(time t) {
    printf("Time: %d : %d : %d\n", t.hours, t.minutes, t.seconds);
    return;
}

time *add_time(time t1, time t2) {
    time  *t3 = (time *) malloc(sizeof(time));
    t3->seconds = t1.seconds + t2.seconds;
    t3->minutes = t1.minutes + t2.minutes;
    t3->hours = t1.hours + t2.hours;
    if(t3->seconds >= 60){
        t3->seconds = t3->seconds - 60;
        t3->minutes++;
    }
    if(t3->minutes >= 60){
        t3->minutes -= 60;
        t3->hours++;
    }
    return t3;
}

int compare_time(time t1, time t2) {
    if(t1.hours > t2.hours)
        return 1;
    else if(t1.hours < t2.hours)
        return -1;
    else if(t1.minutes > t2.minutes)
        return 1;
    else if(t1.minutes < t2.minutes)
        return -1;
    else if(t1.seconds > t2.seconds)
```

```
        return 1;
    else if(t1.seconds < t2.seconds)
        return -1;
    else
        return 0;
}

time *subtract_time(time t1, time t2) {
    time *t3 = (time *) malloc(sizeof(time));
    time larger_t, smaller_t;
    int result = compare_time(t1, t2);
    if(!result) {
        t3->hours = 0;
        t3->minutes = 0;
        t3->seconds = 0;
        return t3;
    }else if(result == 1) {
        larger_t = t1;
        smaller_t = t2;
    }else {
        larger_t = t1;
        smaller_t = t2;
    }
    t3->hours = larger_t.hours - smaller_t.hours;
    if(larger_t.minutes < smaller_t.minutes){
        t3->minutes = 60 - (smaller_t.minutes - larger_t.minutes);
        t3->hours--;
    }else{
        t3->minutes = larger_t.minutes - smaller_t.minutes;
    }
    if(larger_t.seconds < smaller_t.seconds){
        t3->seconds = 60 - (smaller_t.seconds - larger_t.seconds);
        t3->minutes--;
    }else{
        t3->seconds = larger_t.seconds - smaller_t.seconds;
    }
    return t3;
}

int main() {
    int option;
    time t1, t2, *t3;
    t3 = (time *) malloc(sizeof(time));
    read_time(&t1);
    read_time(&t2);
    printf("1. Add dates\n2. Subtract dates\n");
    printf("Enter option: ");
    scanf("%d", &option);
    switch (option) {
        case 1: {
            t3 = add_time(t1, t2);
         break;
     }
        case 2: {
            t3 = subtract_time(t1, t2);
         break;
        }
        default: {
            t3 -> hours = 0;
            t3 -> minutes = 0;
            t3 -> seconds = 0;
         break;
        }
    }
    printf("Result ");
    print_time(*t3);
    return 0;
}
```

Output:



```
 $   gcc -Wall q7.c
 $   ./a.out
Enter hours: 10
Enter minutes: 20
Enter seconds: 50
Enter hours: 5
Enter minutes: 30
Enter seconds: 40
1. Add dates
2. Subtract dates
Enter option: 1
Result Time: 15 : 51 : 30
 $   ./a.out
Enter hours: 10
Enter minutes: 20
Enter seconds: 50
Enter hours: 5
Enter minutes: 30
Enter seconds: 40
1. Add dates
2. Subtract dates
Enter option: 2
Result Time: 4 : 50 : 10
 $   |
```