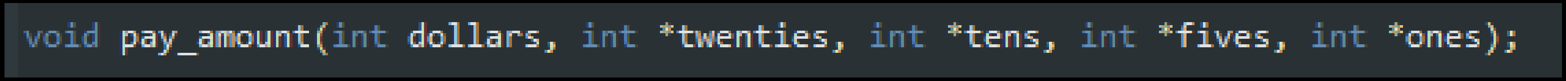
計算機程式語言

教課教授 : 謝東儒

助教 :蔡詠聿、吳品頤

Chapter 11_project 1

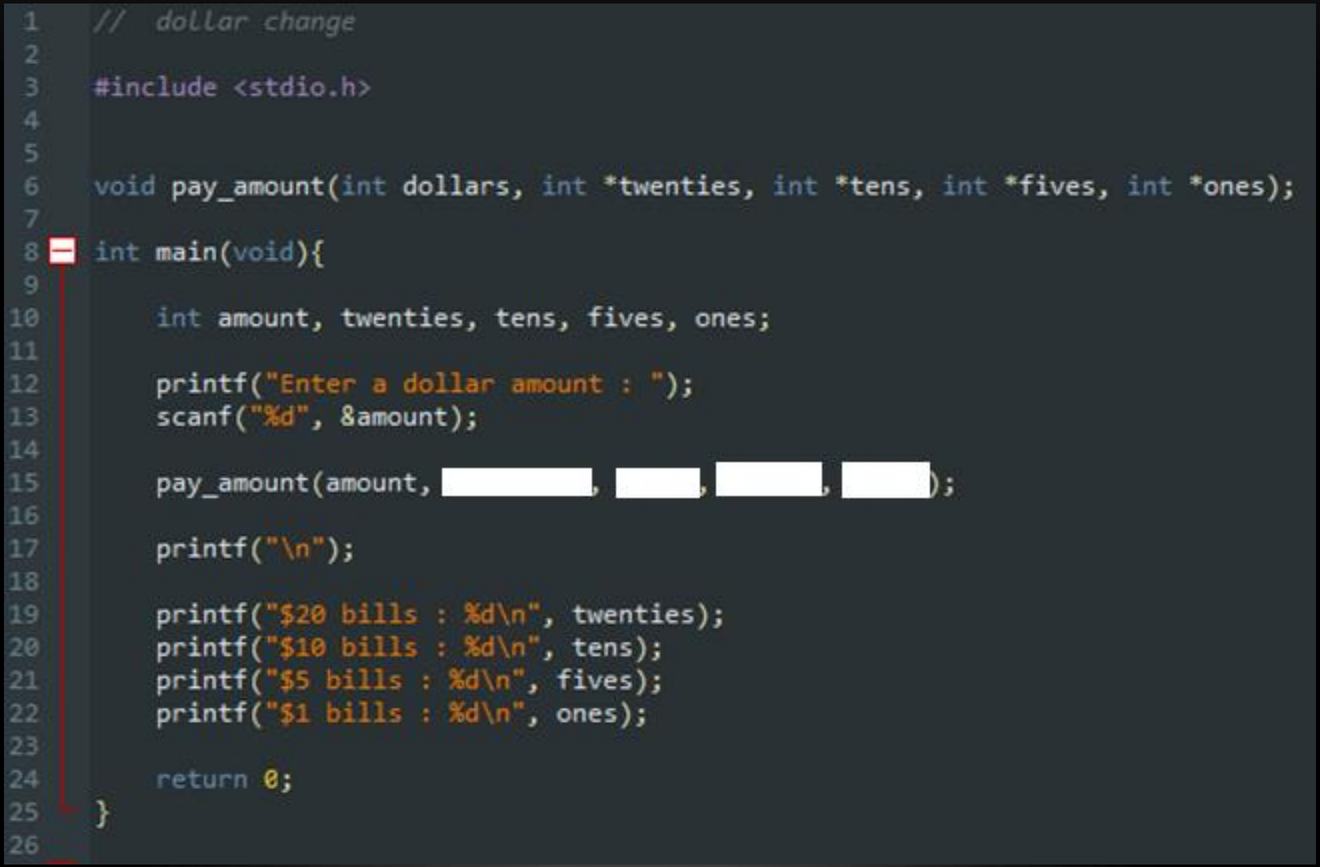
Modify Programming Project 7 from Chapter 2 so that it includes the following function:



```c
void pay_amount(int dollars, int *twenties, int *tens, int *fives, int *ones);
```

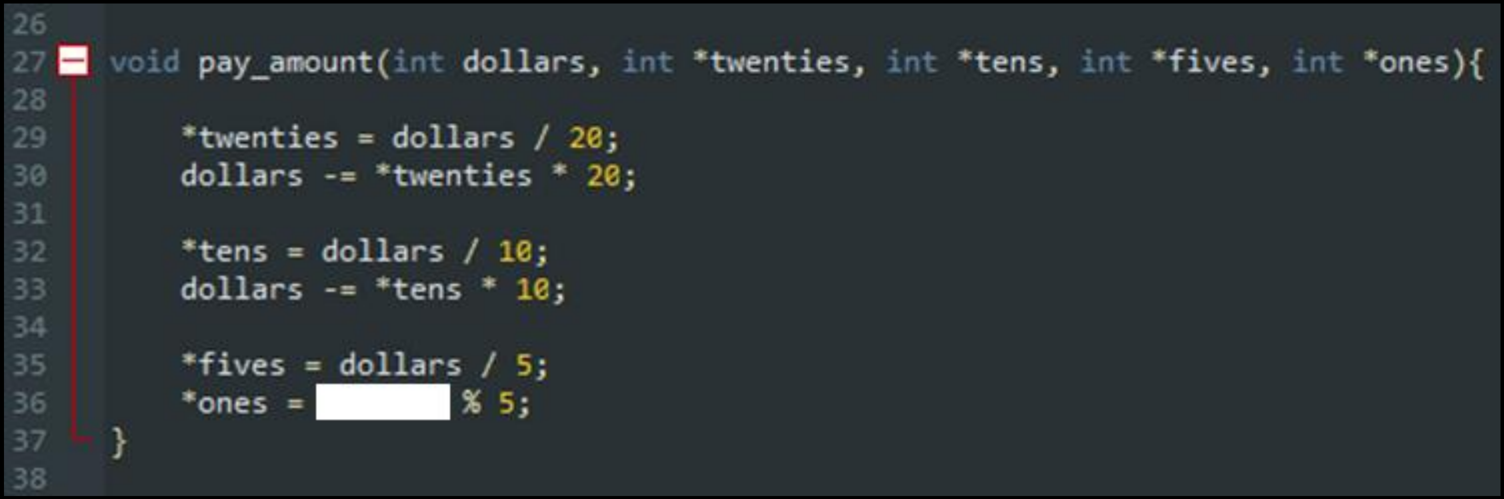
The function determines the smallest number of $20, $10, $5, and $1 bills necessary to pay the amount represented by the dollars parameter. The twenties parameter points to a variable in which the function will store the number of $20 bills required. The tens, fives, and ones parameters are similar.

Solution



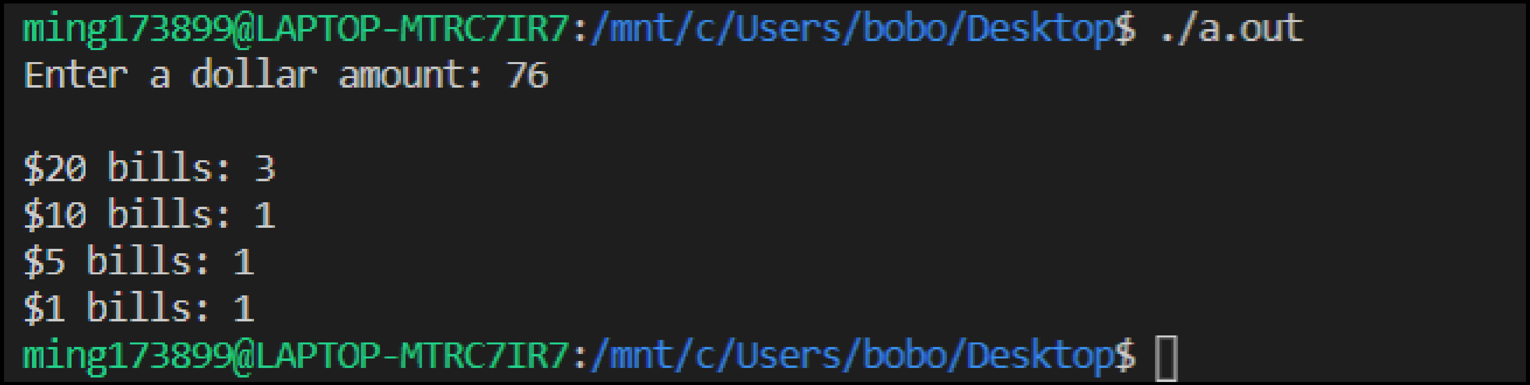
```c
//  dollar change

#include <stdio.h>


void pay_amount(int dollars, int *twenties, int *tens, int *fives, int *ones);

int main(void){

    int amount, twenties, tens, fives, ones;

    printf("Enter a dollar amount : ");
    scanf("%d", &amount);

    pay_amount(amount, [redacted], [redacted], [redacted], [redacted]);

    printf("\n");

    printf("$20 bills : %d\n", twenties);
    printf("$10 bills : %d\n", tens);
    printf("$5 bills : %d\n", fives);
    printf("$1 bills : %d\n", ones);

    return 0;
}

```

Solution



```c

void pay_amount(int dollars, int *twenties, int *tens, int *fives, int *ones){

    *twenties = dollars / 20;
    dollars -= *twenties * 20;

    *tens = dollars / 10;
    dollars -= *tens * 10;

    *fives = dollars / 5;
    *ones =          % 5;
}

```

Example

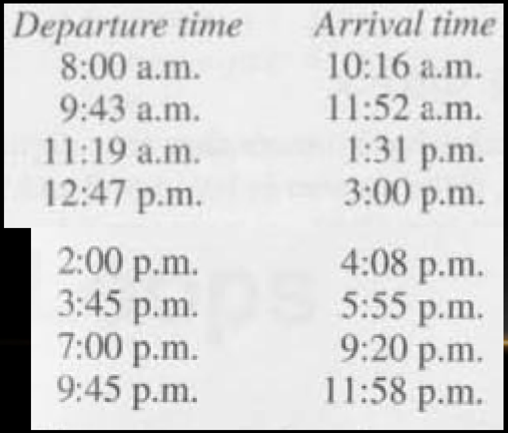

```
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$ ./a.out
Enter a dollar amount: 76

$20 bills: 3
$10 bills: 1
$5 bills: 1
$1 bills: 1
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$
```

Chapter 11_project 2

Modify Programming Project 8 from Chapter 5 so that it includes the following function:

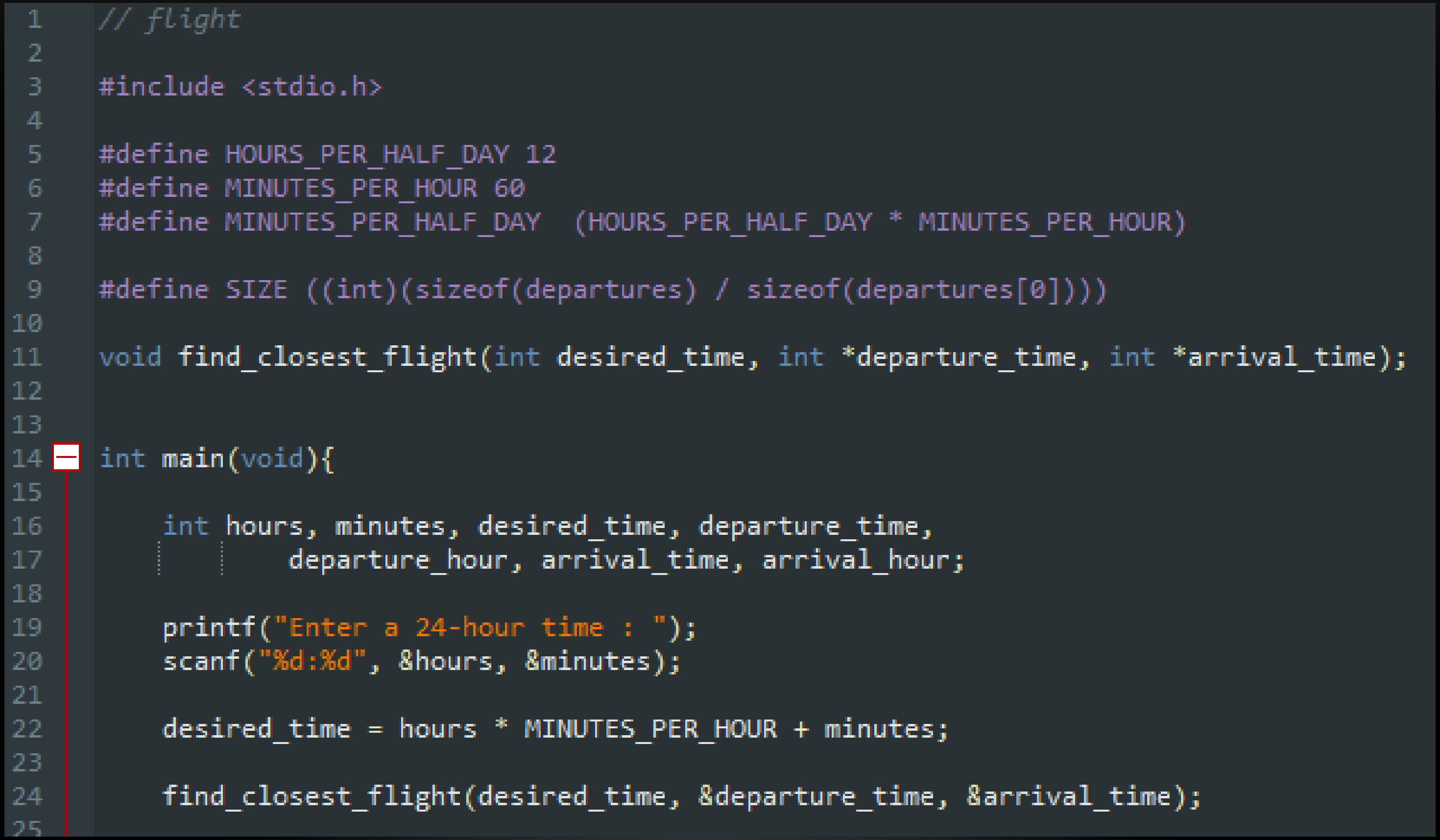
```
void find_closest_flight(int desired_time, int *departure_time, int *arrival_time);
```

This function will find the flight whose departure time is closest to **desired_time** (expressed in minutes since midnight). It will store the departure and arrival times of this flight (also expressed in minutes since midnight) in the variables pointed to by **departure_time** and **arrival_time**, respectively.



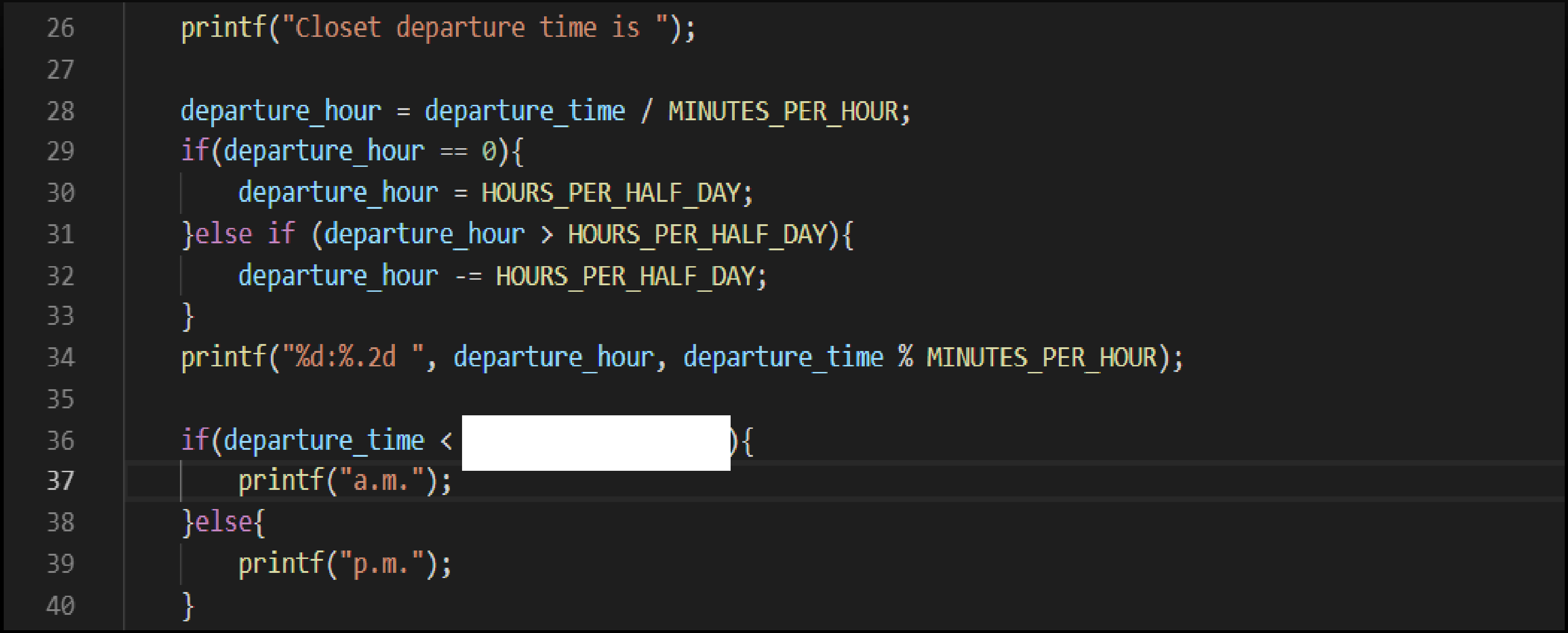
Departure time	*Arrival time*
8:00 a.m.	10:16 a.m.
9:43 a.m.	11:52 a.m.
11:19 a.m.	1:31 p.m.
12:47 p.m.	3:00 p.m.
2:00 p.m.	4:08 p.m.
3:45 p.m.	5:55 p.m.
7:00 p.m.	9:20 p.m.
9:45 p.m.	11:58 p.m.

Solution



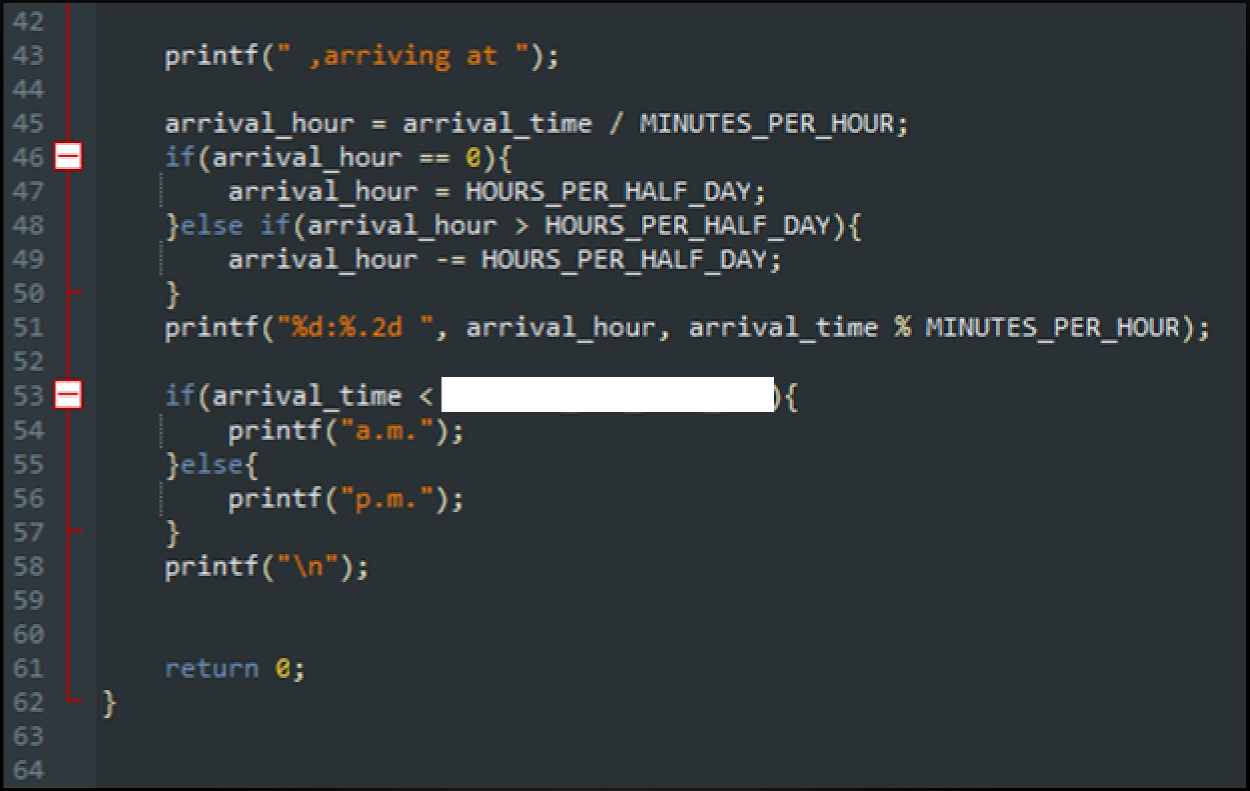
```
// flight

#include <stdio.h>

#define HOURS_PER_HALF_DAY 12
#define MINUTES_PER_HOUR 60
#define MINUTES_PER_HALF_DAY  (HOURS_PER_HALF_DAY * MINUTES_PER_HOUR)

#define SIZE ((int)(sizeof(departures) / sizeof(departures[0])))

void find_closest_flight(int desired_time, int *departure_time, int *arrival_time);


int main(void){

    int hours, minutes, desired_time, departure_time,
            departure_hour, arrival_time, arrival_hour;

    printf("Enter a 24-hour time : ");
    scanf("%d:%d", &hours, &minutes);

    desired_time = hours * MINUTES_PER_HOUR + minutes;

    find_closest_flight(desired_time, &departure_time, &arrival_time);
```

Solution



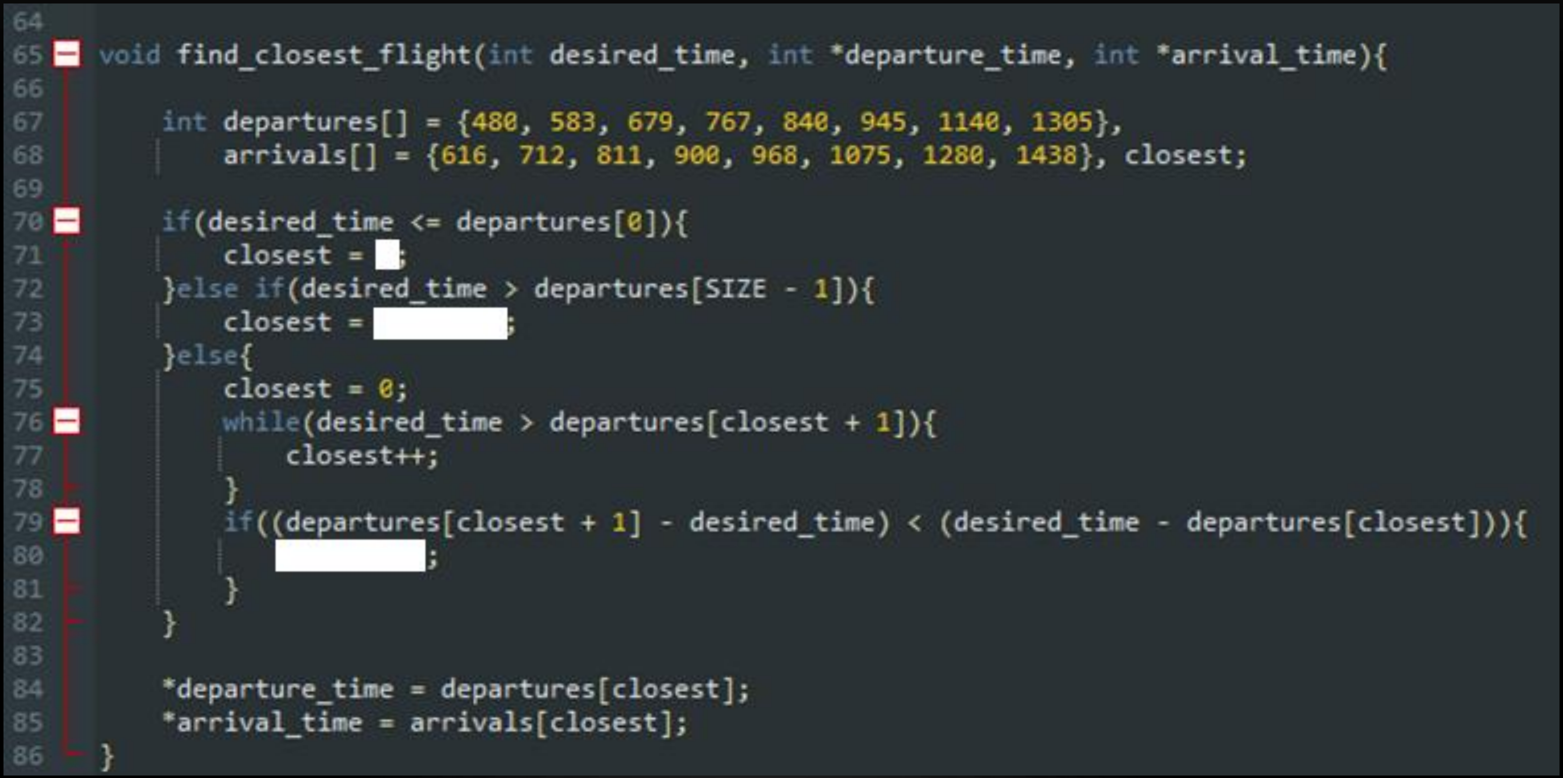
```c
printf("Closet departure time is ");

departure_hour = departure_time / MINUTES_PER_HOUR;
if(departure_hour == 0){
    departure_hour = HOURS_PER_HALF_DAY;
}else if (departure_hour > HOURS_PER_HALF_DAY){
    departure_hour -= HOURS_PER_HALF_DAY;
}
printf("%d:%.2d ", departure_hour, departure_time % MINUTES_PER_HOUR);

if(departure_time < [redacted]){
    printf("a.m.");
}else{
    printf("p.m.");
}
```

Solution



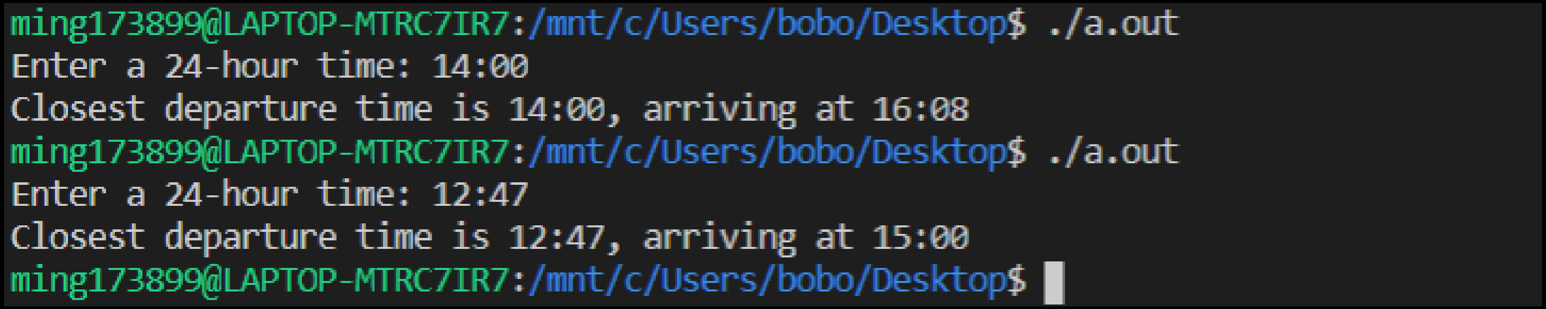
```c

    printf(" ,arriving at ");

    arrival_hour = arrival_time / MINUTES_PER_HOUR;
    if(arrival_hour == 0){
        arrival_hour = HOURS_PER_HALF_DAY;
    }else if(arrival_hour > HOURS_PER_HALF_DAY){
        arrival_hour -= HOURS_PER_HALF_DAY;
    }
    printf("%d:%.2d ", arrival_hour, arrival_time % MINUTES_PER_HOUR);

    if(arrival_time <                     ){
        printf("a.m.");
    }else{
        printf("p.m.");
    }
    printf("\n");


    return 0;
}


```

Solution



```
void find_closest_flight(int desired_time, int *departure_time, int *arrival_time){

    int departures[] = {480, 583, 679, 767, 840, 945, 1140, 1305},
        arrivals[] = {616, 712, 811, 900, 968, 1075, 1280, 1438}, closest;

    if(desired_time <= departures[0]){
        closest = ;
    }else if(desired_time > departures[SIZE - 1]){
        closest = ;
    }else{
        closest = 0;
        while(desired_time > departures[closest + 1]){
            closest++;
        }
        if((departures[closest + 1] - desired_time) < (desired_time - departures[closest])){
            ;
        }
    }

    *departure_time = departures[closest];
    *arrival_time = arrivals[closest];
}
```

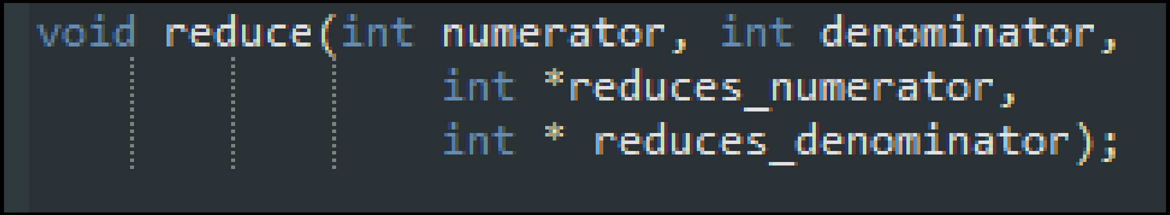
Example



```
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$ ./a.out
Enter a 24-hour time: 14:00
Closest departure time is 14:00, arriving at 16:08
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$ ./a.out
Enter a 24-hour time: 12:47
Closest departure time is 12:47, arriving at 15:00
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$
```

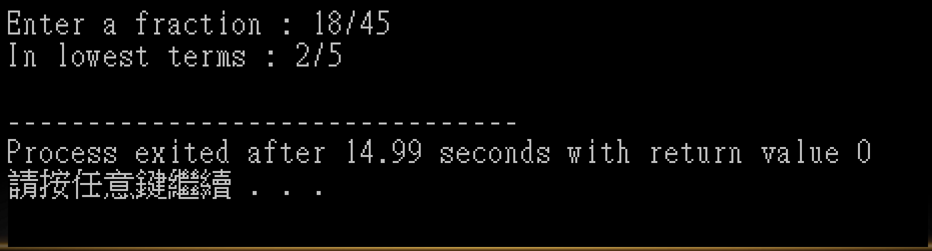
Chapter 11_project 3

Modify Programming Project 3 from Chapter 6 so that it includes the following function:



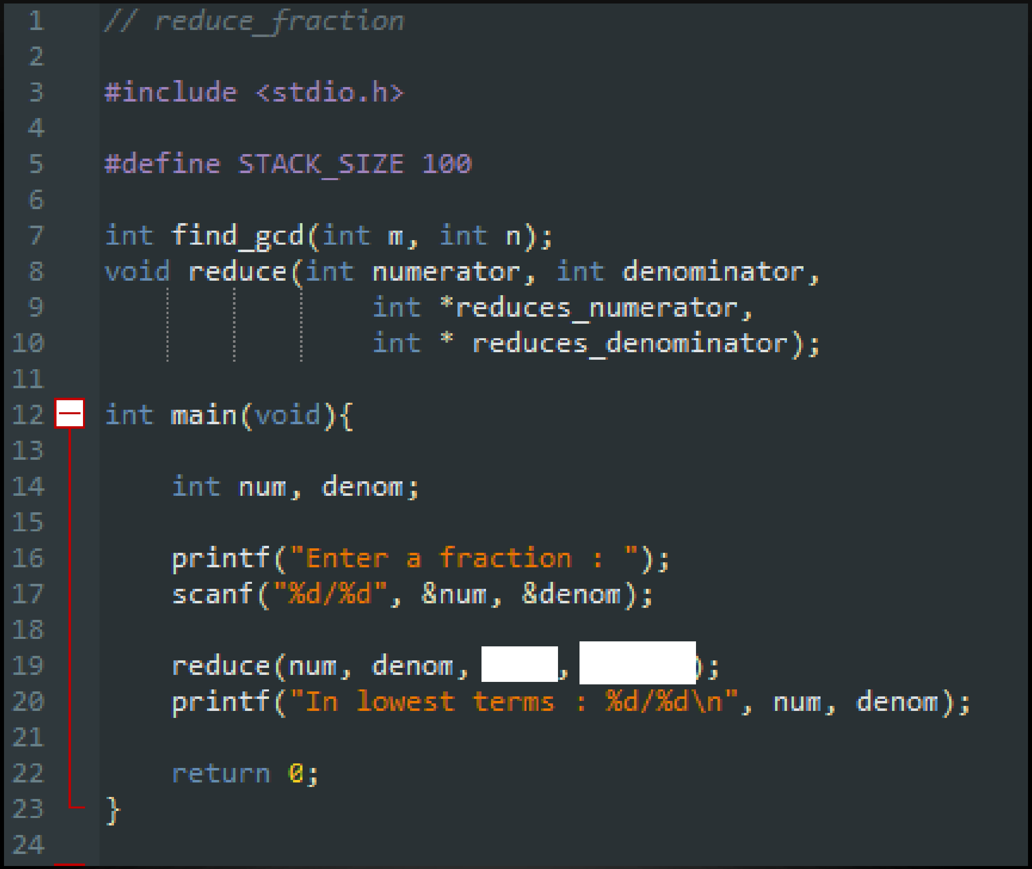
```
void reduce(int numerator, int denominator,
                int *reduces_numerator,
                int * reduces_denominator);
```

numerator and **denominator** are the numerator and denominator of a fraction, **reduced_numerator** and **reduced_ denominator** are pointers to variables in which the function will store the numerator and denominator of the fraction once it has been reduced to lowest terms.



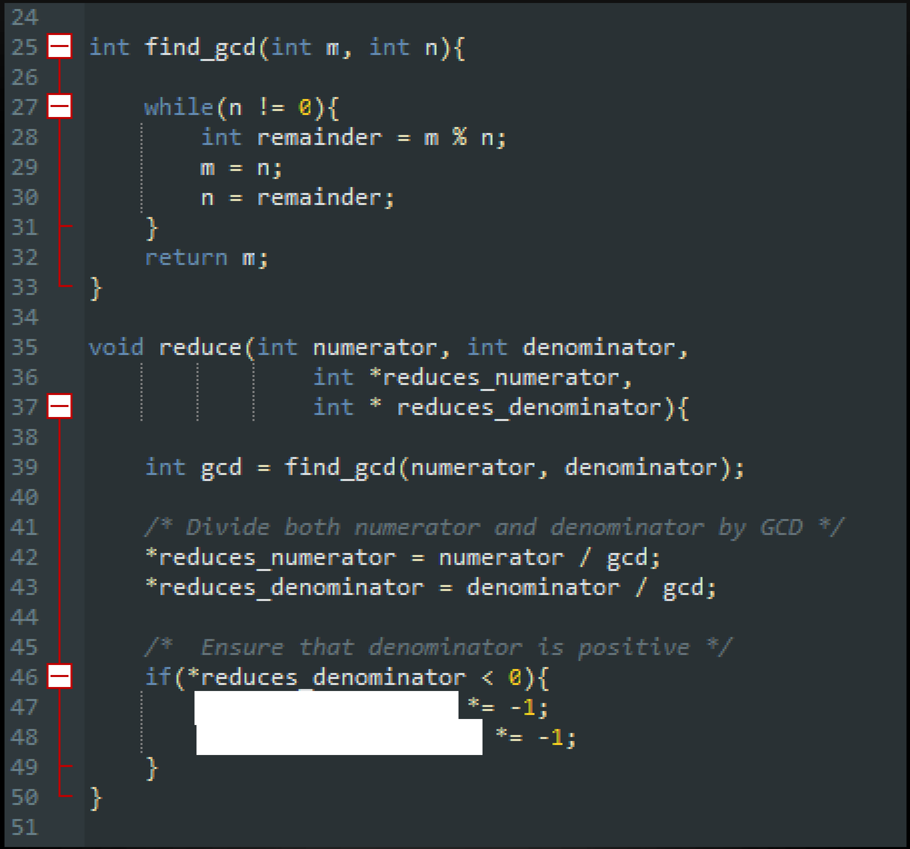
```
Enter a fraction : 18/45
In lowest terms : 2/5

--------------------------------
Process exited after 14.99 seconds with return value 0
請按任意鍵繼續 . . .
```

Solution



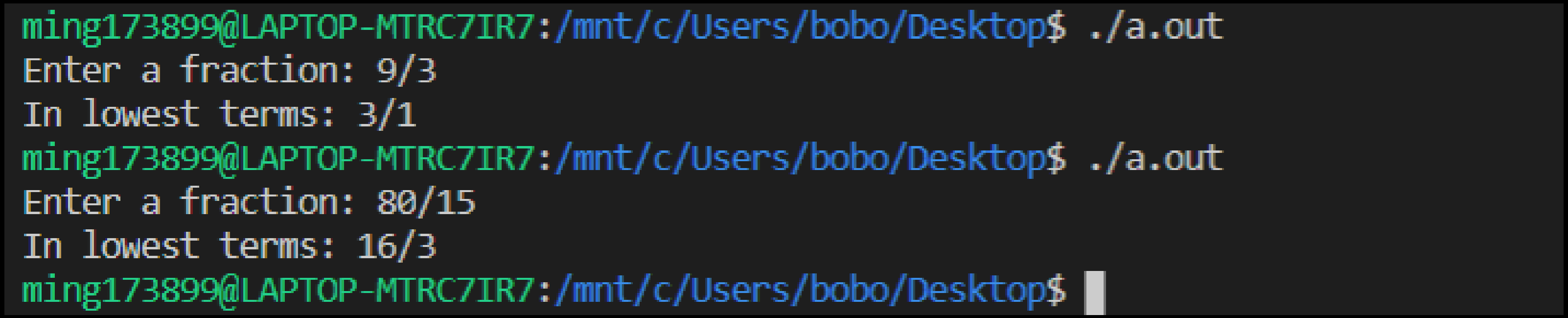
```c
// reduce_fraction

#include <stdio.h>

#define STACK_SIZE 100

int find_gcd(int m, int n);
void reduce(int numerator, int denominator,
            int *reduces_numerator,
            int * reduces_denominator);

int main(void){

    int num, denom;

    printf("Enter a fraction : ");
    scanf("%d/%d", &num, &denom);

    reduce(num, denom, [redacted], [redacted]);
    printf("In lowest terms : %d/%d\n", num, denom);

    return 0;
}

```

Solution



```

int find_gcd(int m, int n){

    while(n != 0){
        int remainder = m % n;
        m = n;
        n = remainder;
    }
    return m;
}

void reduce(int numerator, int denominator,
                int *reduces_numerator,
                int * reduces_denominator){

    int gcd = find_gcd(numerator, denominator);

    /* Divide both numerator and denominator by GCD */
    *reduces_numerator = numerator / gcd;
    *reduces_denominator = denominator / gcd;

    /*  Ensure that denominator is positive */
    if(*reduces_denominator < 0){
        [redacted] *= -1;
        [redacted] *= -1;
    }
}

```

Example

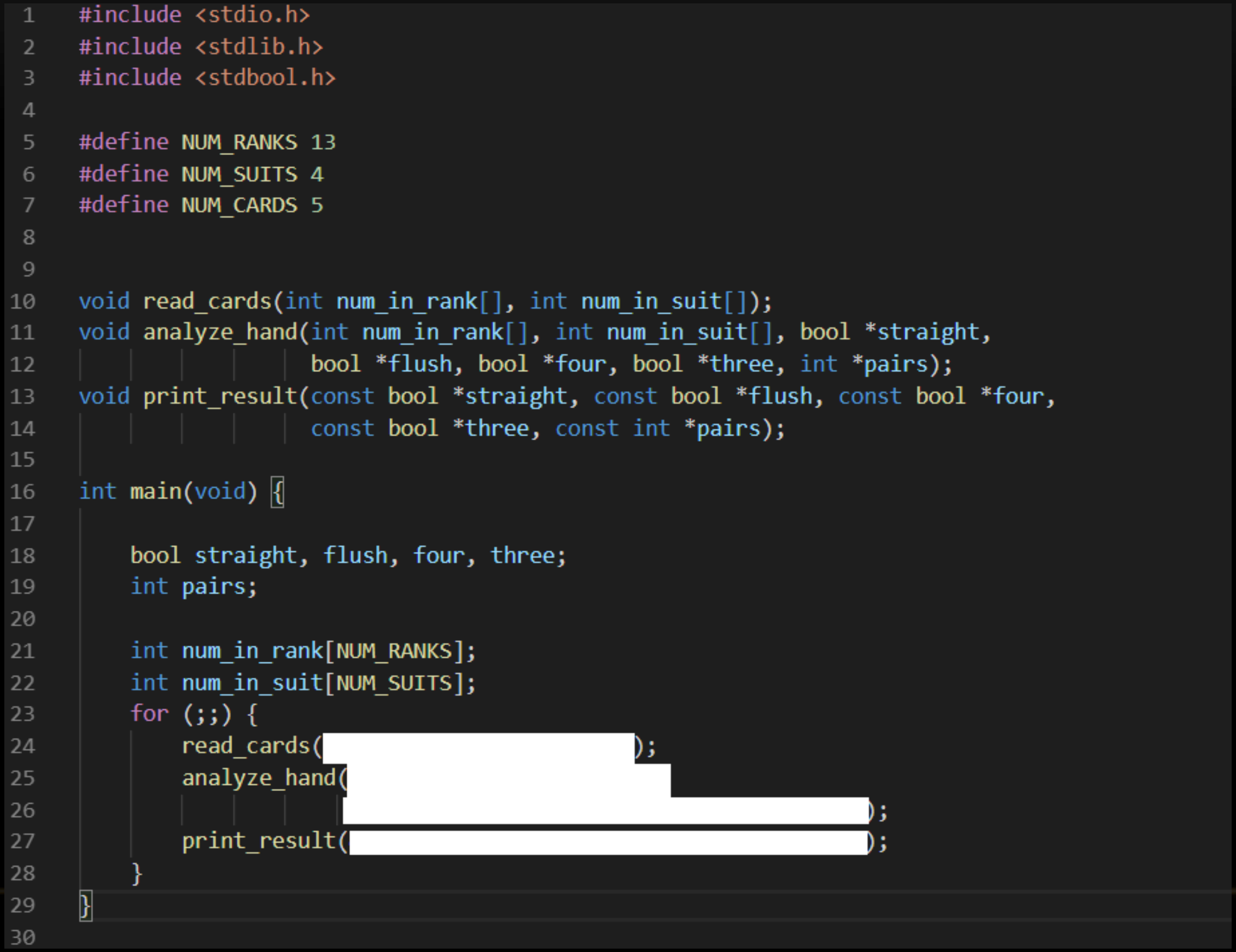


```
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$ ./a.out
Enter a fraction: 9/3
In lowest terms: 3/1
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$ ./a.out
Enter a fraction: 80/15
In lowest terms: 16/3
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$
```

Chapter 11_project 4

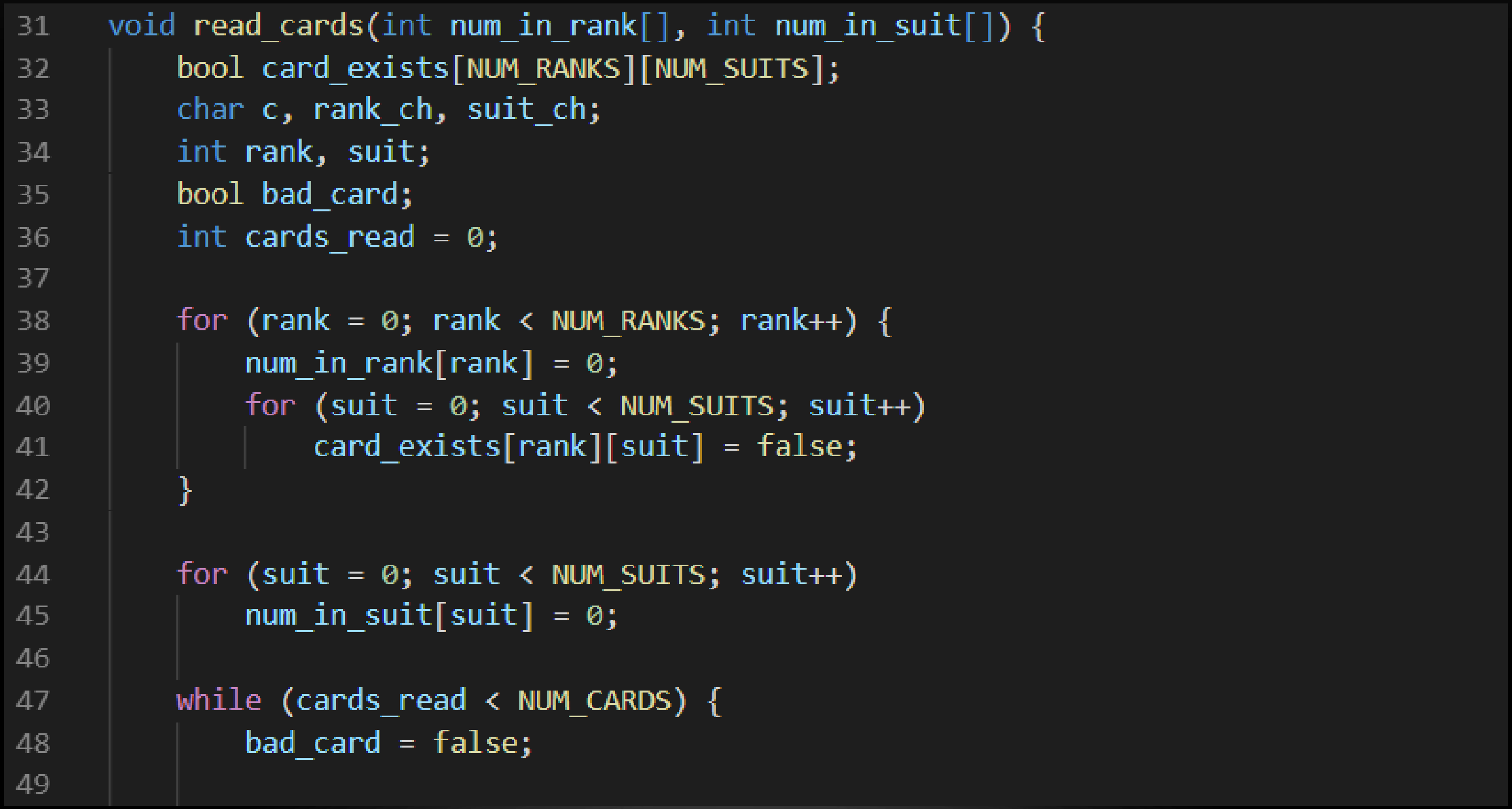
Modify the poker.c program of Section 10.5 by moving all external variables into main and nodifying functions so that they communicate by passing arguments. The analyze_hand function needs to change the straight, flush, four, three, and pairs variables, so it will have to be passed pointers to those variables.

Solution



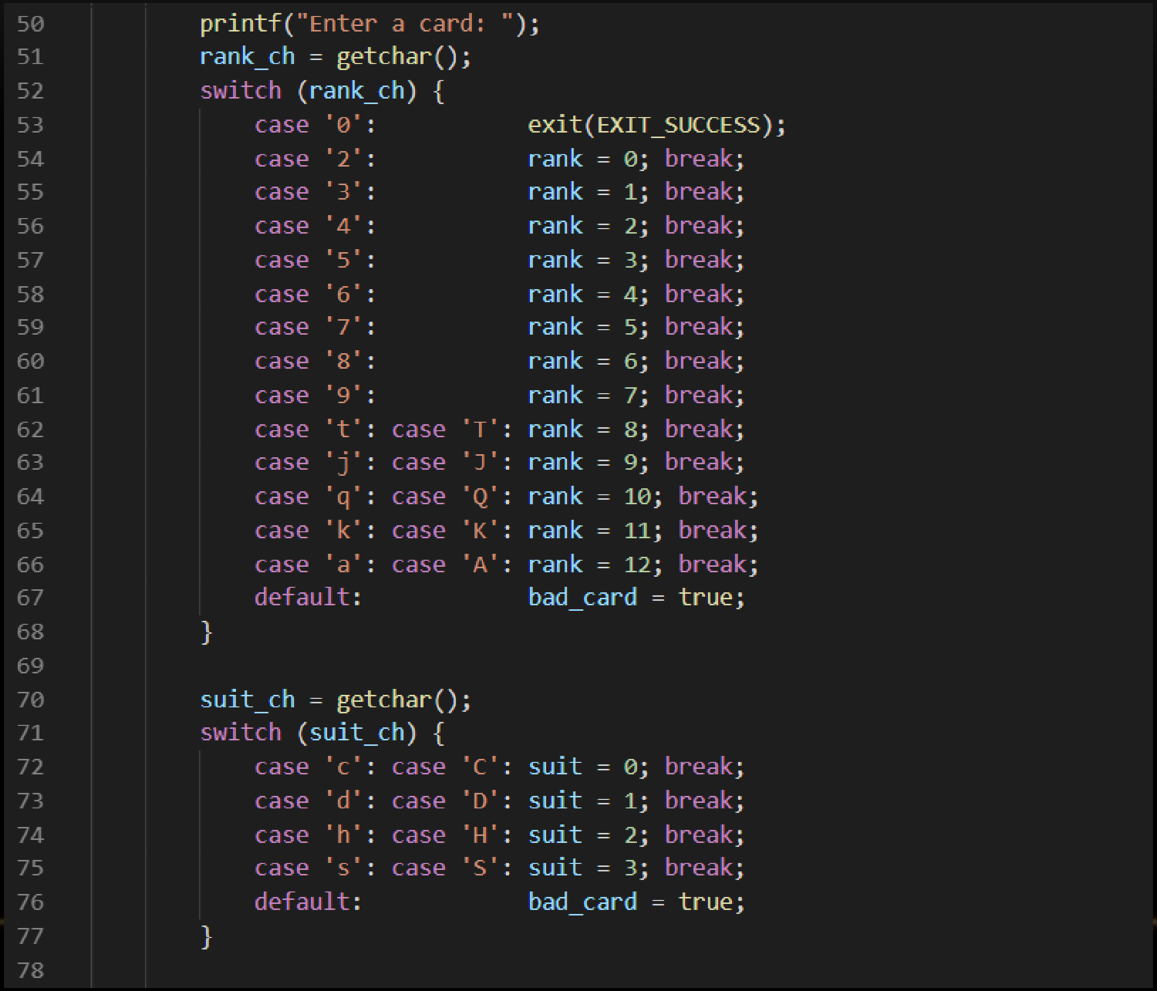
```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define NUM_RANKS 13
#define NUM_SUITS 4
#define NUM_CARDS 5


void read_cards(int num_in_rank[], int num_in_suit[]);
void analyze_hand(int num_in_rank[], int num_in_suit[], bool *straight,
                  bool *flush, bool *four, bool *three, int *pairs);
void print_result(const bool *straight, const bool *flush, const bool *four,
                  const bool *three, const int *pairs);

int main(void) {

    bool straight, flush, four, three;
    int pairs;

    int num_in_rank[NUM_RANKS];
    int num_in_suit[NUM_SUITS];
    for (;;) {
        read_cards([redacted]);
        analyze_hand([redacted]);
        print_result([redacted]);
    }
}

```

Solution



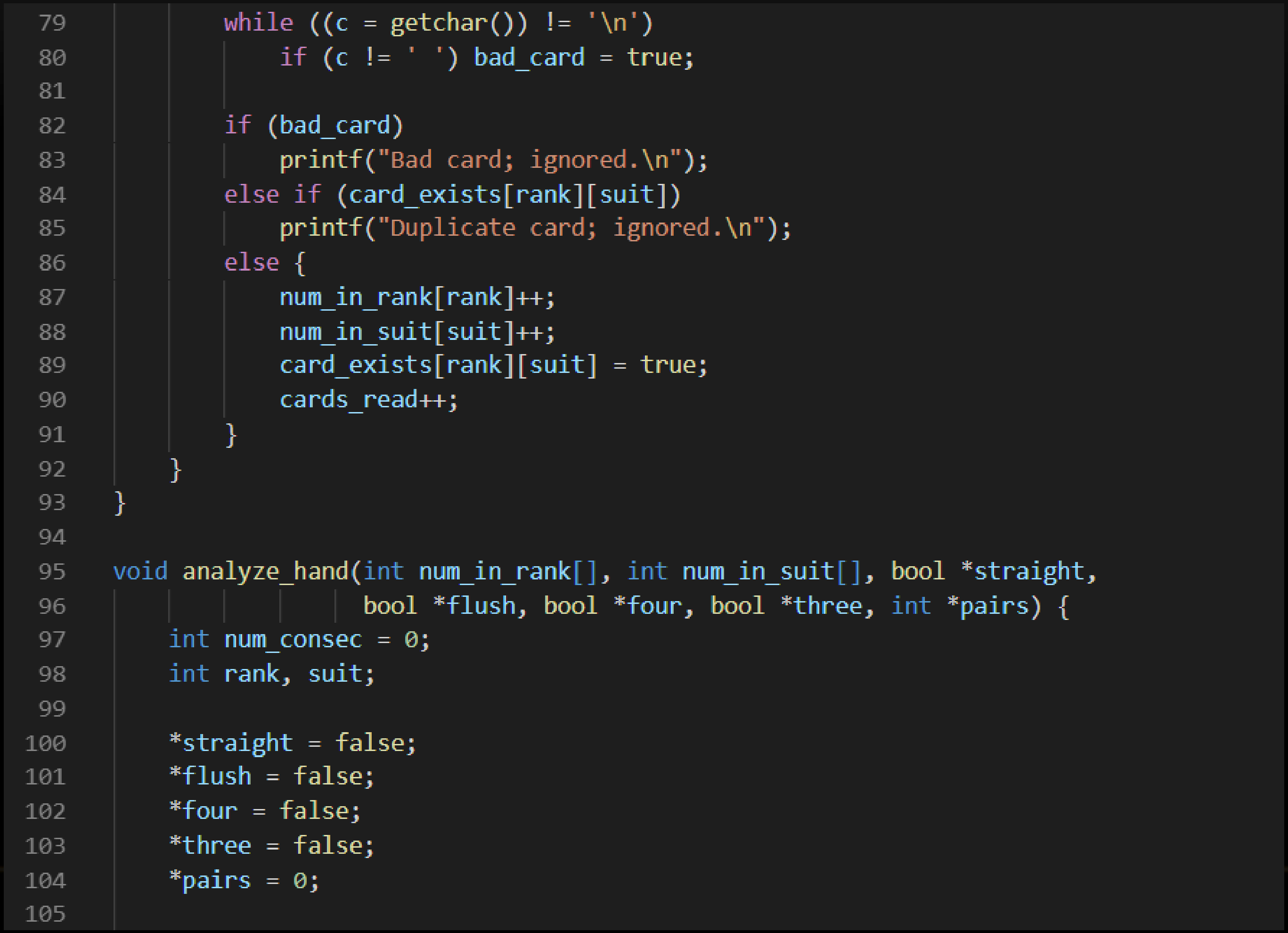
```c
void read_cards(int num_in_rank[], int num_in_suit[]) {
    bool card_exists[NUM_RANKS][NUM_SUITS];
    char c, rank_ch, suit_ch;
    int rank, suit;
    bool bad_card;
    int cards_read = 0;

    for (rank = 0; rank < NUM_RANKS; rank++) {
        num_in_rank[rank] = 0;
        for (suit = 0; suit < NUM_SUITS; suit++)
            card_exists[rank][suit] = false;
    }

    for (suit = 0; suit < NUM_SUITS; suit++)
        num_in_suit[suit] = 0;

    while (cards_read < NUM_CARDS) {
        bad_card = false;

```

Solution



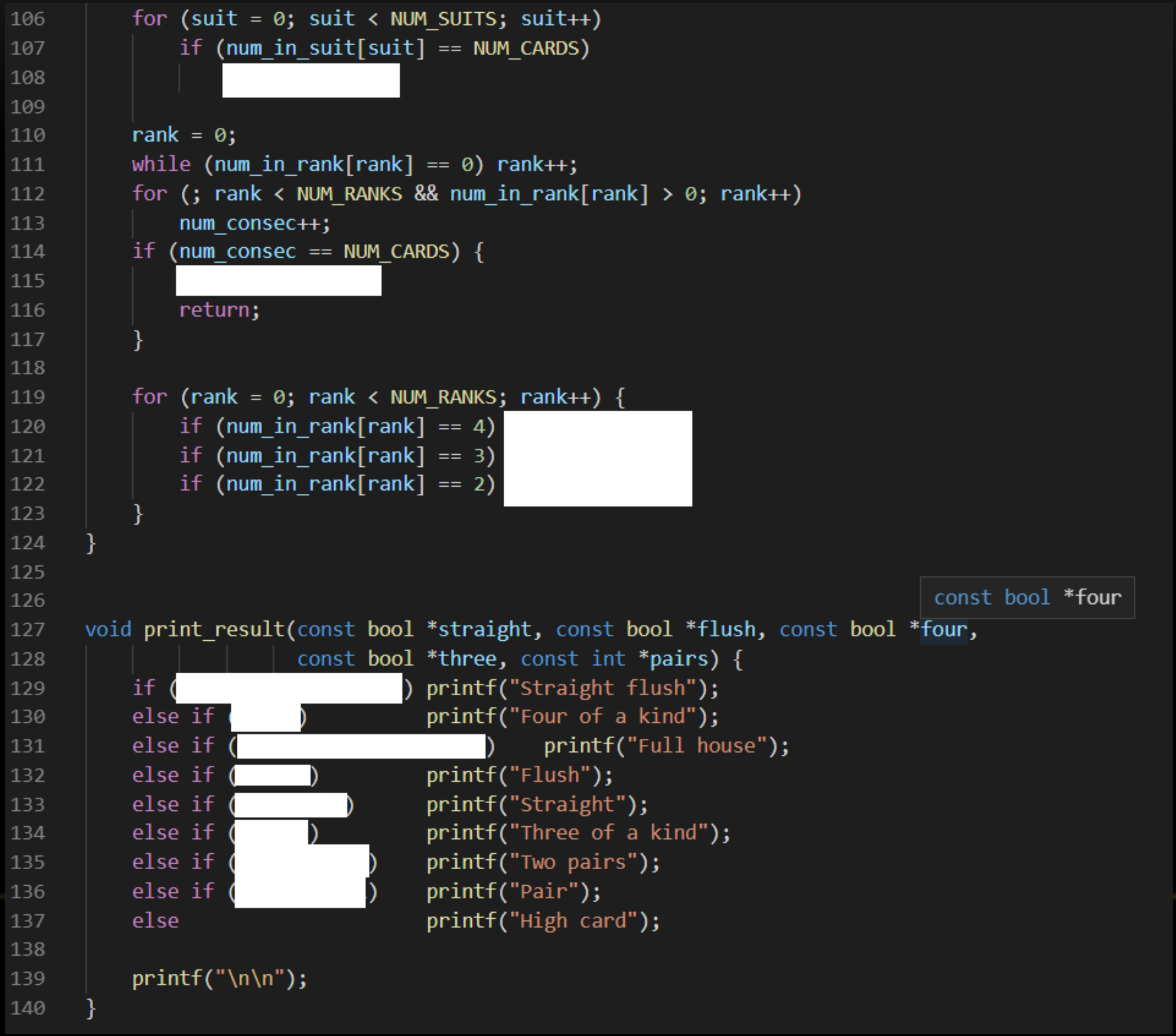
```c
        printf("Enter a card: ");
        rank_ch = getchar();
        switch (rank_ch) {
            case '0':           exit(EXIT_SUCCESS);
            case '2':           rank = 0; break;
            case '3':           rank = 1; break;
            case '4':           rank = 2; break;
            case '5':           rank = 3; break;
            case '6':           rank = 4; break;
            case '7':           rank = 5; break;
            case '8':           rank = 6; break;
            case '9':           rank = 7; break;
            case 't': case 'T': rank = 8; break;
            case 'j': case 'J': rank = 9; break;
            case 'q': case 'Q': rank = 10; break;
            case 'k': case 'K': rank = 11; break;
            case 'a': case 'A': rank = 12; break;
            default:            bad_card = true;
        }

        suit_ch = getchar();
        switch (suit_ch) {
            case 'c': case 'C': suit = 0; break;
            case 'd': case 'D': suit = 1; break;
            case 'h': case 'H': suit = 2; break;
            case 's': case 'S': suit = 3; break;
            default:            bad_card = true;
        }

```

Solution



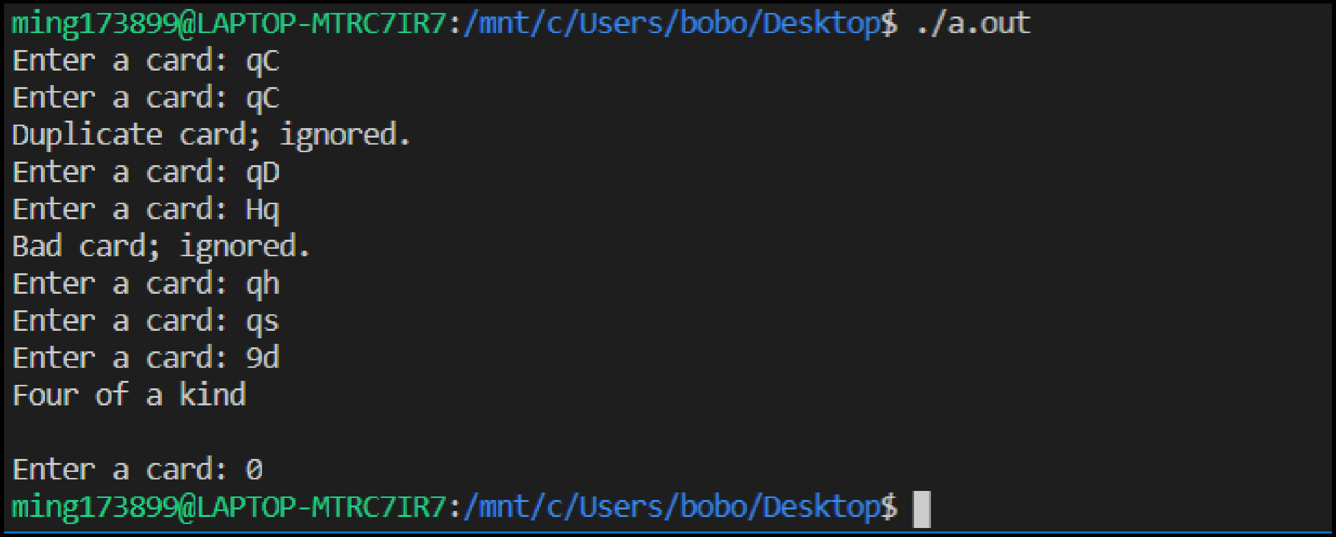
```c
        while ((c = getchar()) != '\n')
            if (c != ' ') bad_card = true;

        if (bad_card)
            printf("Bad card; ignored.\n");
        else if (card_exists[rank][suit])
            printf("Duplicate card; ignored.\n");
        else {
            num_in_rank[rank]++;
            num_in_suit[suit]++;
            card_exists[rank][suit] = true;
            cards_read++;
        }
    }
}

void analyze_hand(int num_in_rank[], int num_in_suit[], bool *straight,
                  bool *flush, bool *four, bool *three, int *pairs) {
    int num_consec = 0;
    int rank, suit;

    *straight = false;
    *flush = false;
    *four = false;
    *three = false;
    *pairs = 0;

```

Solution



```
    for (suit = 0; suit < NUM_SUITS; suit++)
        if (num_in_suit[suit] == NUM_CARDS)
            [redacted]

    rank = 0;
    while (num_in_rank[rank] == 0) rank++;
    for (; rank < NUM_RANKS && num_in_rank[rank] > 0; rank++)
        num_consec++;
    if (num_consec == NUM_CARDS) {
        [redacted]
        return;
    }

    for (rank = 0; rank < NUM_RANKS; rank++) {
        if (num_in_rank[rank] == 4) [redacted]
        if (num_in_rank[rank] == 3)
        if (num_in_rank[rank] == 2)
    }
}


void print_result(const bool *straight, const bool *flush, const bool *four,
                  const bool *three, const int *pairs) {
    if ([redacted]) printf("Straight flush");
    else if ([redacted])            printf("Four of a kind");
    else if ([redacted])     printf("Full house");
    else if ([redacted])            printf("Flush");
    else if ([redacted])            printf("Straight");
    else if ([redacted])            printf("Three of a kind");
    else if ([redacted])            printf("Two pairs");
    else if ([redacted])            printf("Pair");
    else                            printf("High card");

    printf("\n\n");
}
```

Example



```
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$ ./a.out
Enter a card: qC
Enter a card: qC
Duplicate card; ignored.
Enter a card: qD
Enter a card: Hq
Bad card; ignored.
Enter a card: qh
Enter a card: qs
Enter a card: 9d
Four of a kind

Enter a card: 0
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$
```