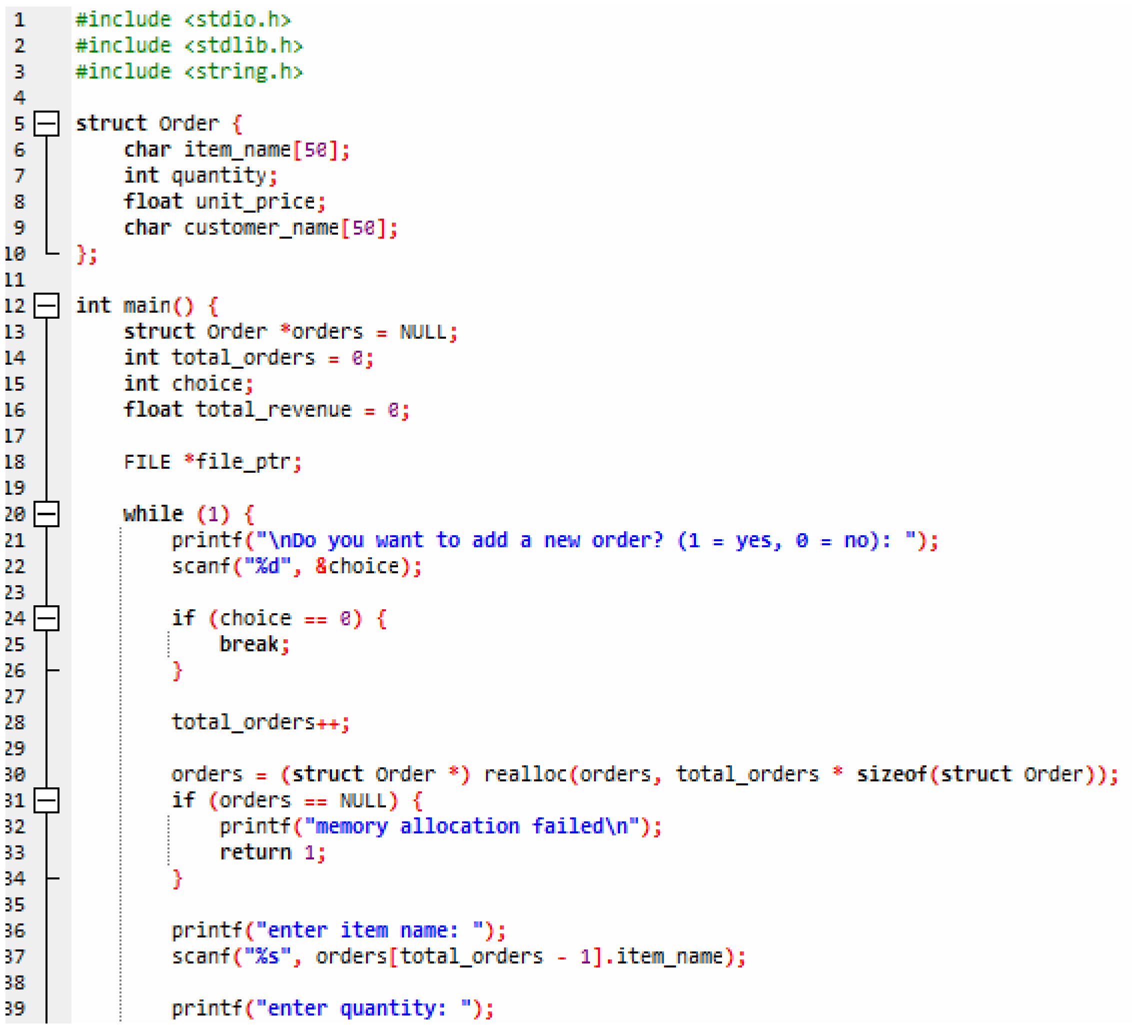
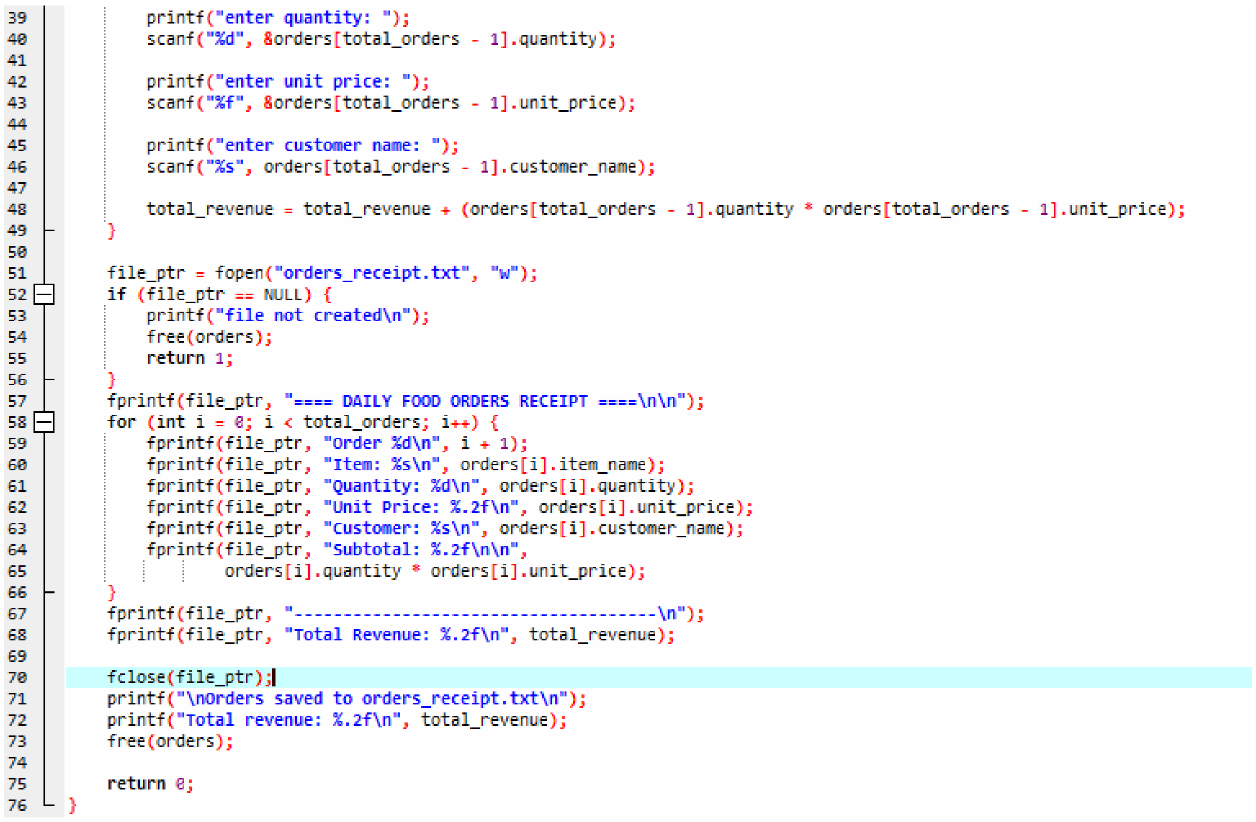
25K-2038 (LAB MANUAL 12)

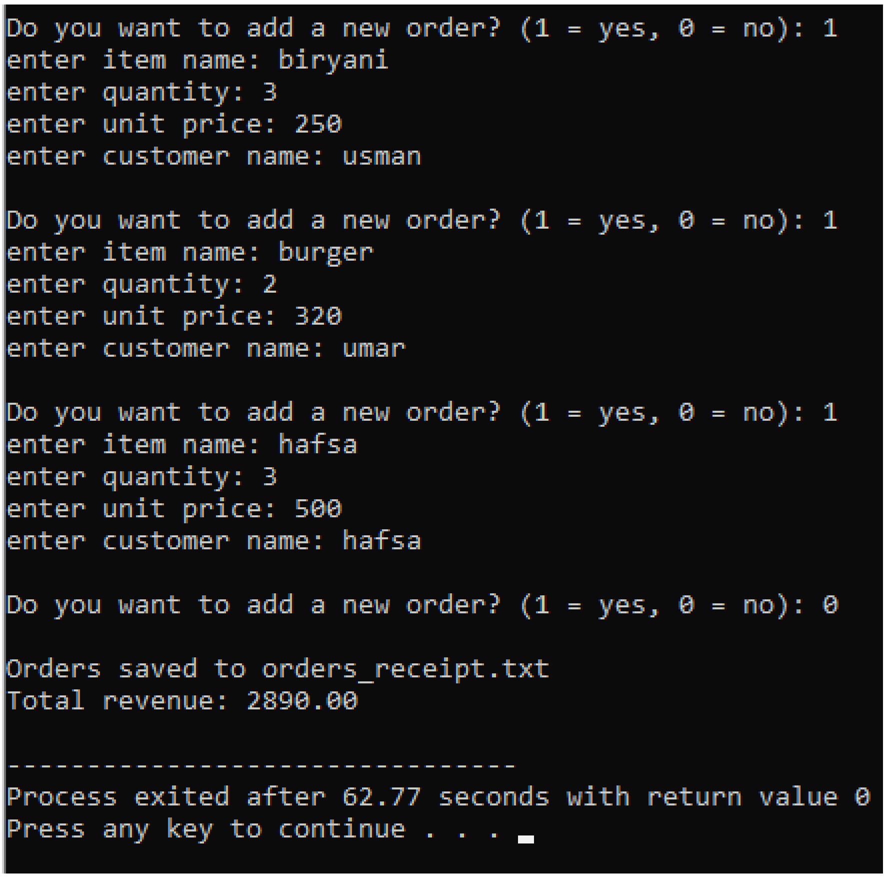
Question #1



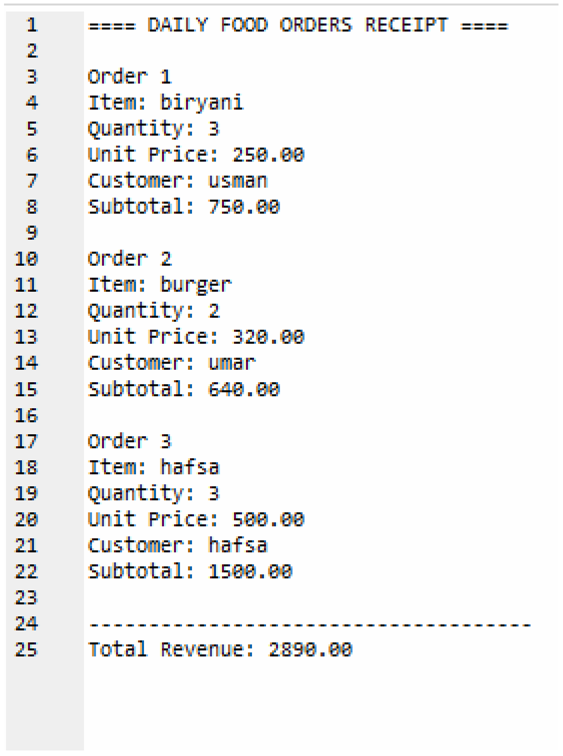
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Order {
    char item_name[50];
    int quantity;
    float unit_price;
    char customer_name[50];
};

int main() {
    struct Order *orders = NULL;
    int total_orders = 0;
    int choice;
    float total_revenue = 0;

    FILE *file_ptr;

    while (1) {
        printf("\nDo you want to add a new order? (1 = yes, 0 = no): ");
        scanf("%d", &choice);

        if (choice == 0) {
            break;
        }

        total_orders++;

        orders = (struct Order *) realloc(orders, total_orders * sizeof(struct Order));
        if (orders == NULL) {
            printf("memory allocation failed\n");
            return 1;
        }

        printf("enter item name: ");
        scanf("%s", orders[total_orders - 1].item_name);

        printf("enter quantity: ");
```



```c
        printf("enter quantity: ");
        scanf("%d", &orders[total_orders - 1].quantity);

        printf("enter unit price: ");
        scanf("%f", &orders[total_orders - 1].unit_price);

        printf("enter customer name: ");
        scanf("%s", orders[total_orders - 1].customer_name);

        total_revenue = total_revenue + (orders[total_orders - 1].quantity * orders[total_orders - 1].unit_price);
    }

    file_ptr = fopen("orders_receipt.txt", "w");
    if (file_ptr == NULL) {
        printf("file not created\n");
        free(orders);
        return 1;
    }
    fprintf(file_ptr, "==== DAILY FOOD ORDERS RECEIPT ====\n\n");
    for (int i = 0; i < total_orders; i++) {
        fprintf(file_ptr, "Order %d\n", i + 1);
        fprintf(file_ptr, "Item: %s\n", orders[i].item_name);
        fprintf(file_ptr, "Quantity: %d\n", orders[i].quantity);
        fprintf(file_ptr, "Unit Price: %.2f\n", orders[i].unit_price);
        fprintf(file_ptr, "Customer: %s\n", orders[i].customer_name);
        fprintf(file_ptr, "Subtotal: %.2f\n\n",
                orders[i].quantity * orders[i].unit_price);
    }
    fprintf(file_ptr, "------------------------------------\n");
    fprintf(file_ptr, "Total Revenue: %.2f\n", total_revenue);

    fclose(file_ptr);
    printf("\nOrders saved to orders_receipt.txt\n");
    printf("Total revenue: %.2f\n", total_revenue);
    free(orders);

    return 0;
}
```

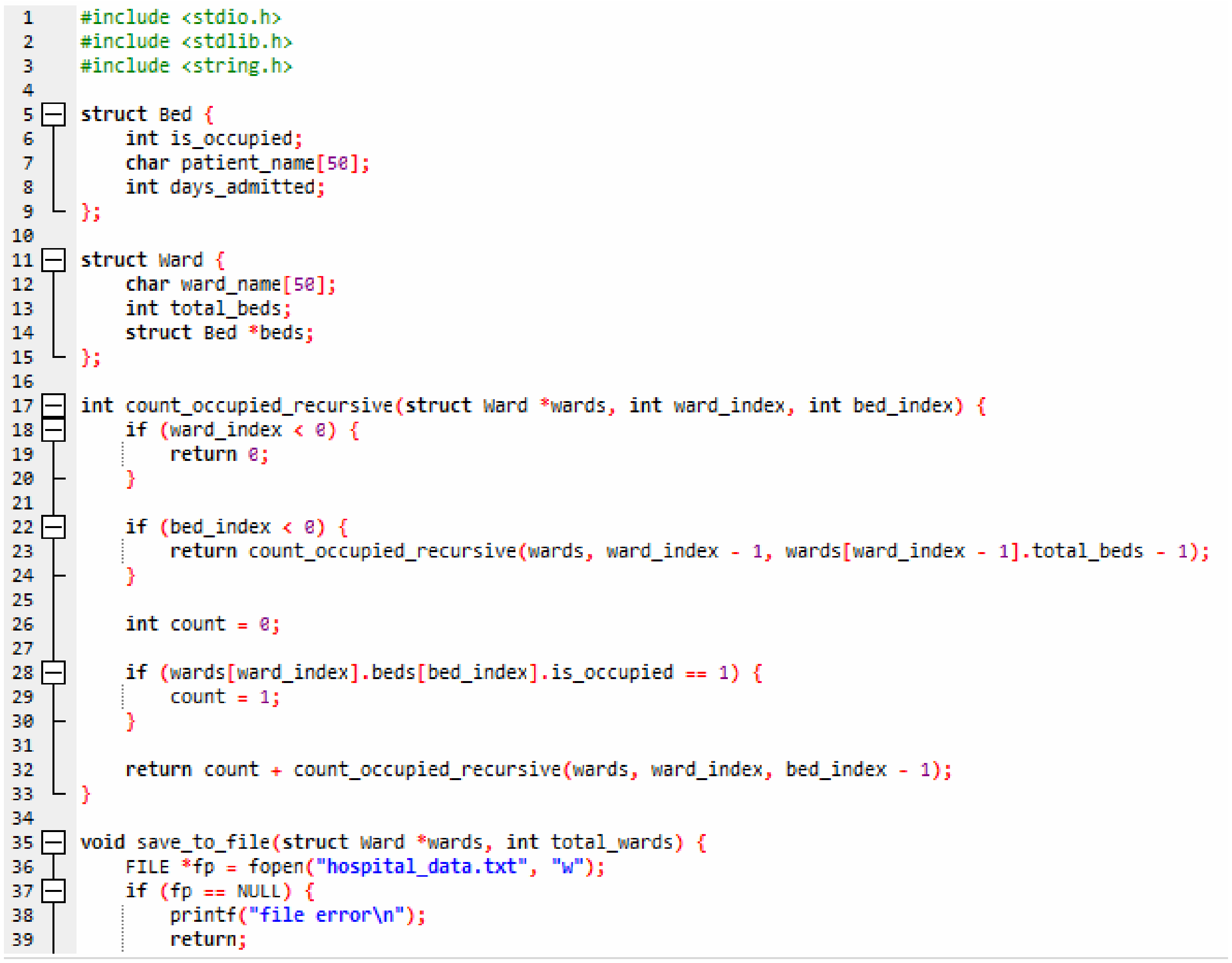


```
Do you want to add a new order? (1 = yes, 0 = no): 1
enter item name: biryani
enter quantity: 3
enter unit price: 250
enter customer name: usman

Do you want to add a new order? (1 = yes, 0 = no): 1
enter item name: burger
enter quantity: 2
enter unit price: 320
enter customer name: umar

Do you want to add a new order? (1 = yes, 0 = no): 1
enter item name: hafsa
enter quantity: 3
enter unit price: 500
enter customer name: hafsa

Do you want to add a new order? (1 = yes, 0 = no): 0

Orders saved to orders_receipt.txt
Total revenue: 2890.00

--------------------------------
Process exited after 62.77 seconds with return value 0
Press any key to continue . . .
```

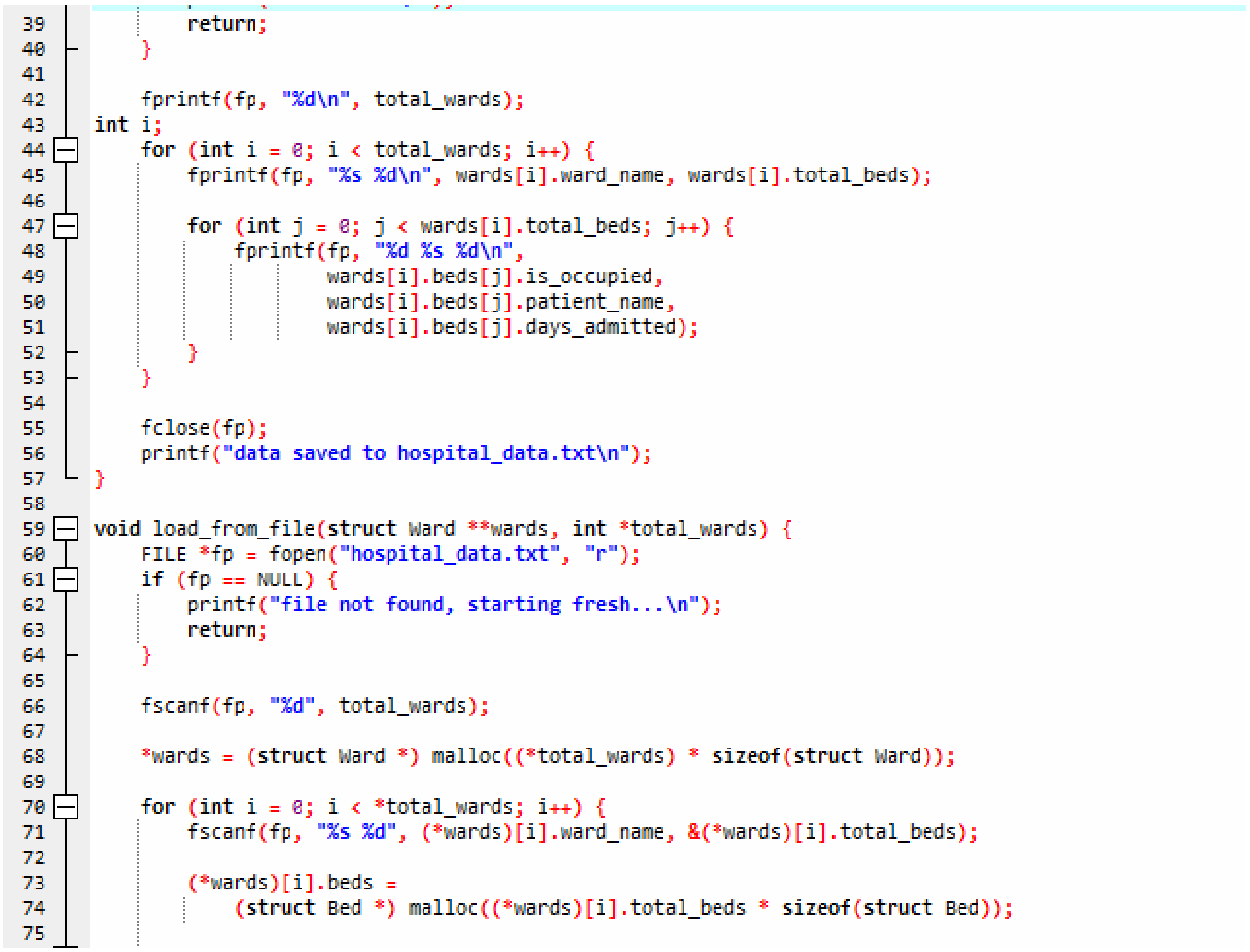


```
==== DAILY FOOD ORDERS RECEIPT ====

Order 1
Item: biryani
Quantity: 3
Unit Price: 250.00
Customer: usman
Subtotal: 750.00

Order 2
Item: burger
Quantity: 2
Unit Price: 320.00
Customer: umar
Subtotal: 640.00

Order 3
Item: hafsa
Quantity: 3
Unit Price: 500.00
Customer: hafsa
Subtotal: 1500.00

------------------------------------
Total Revenue: 2890.00
```

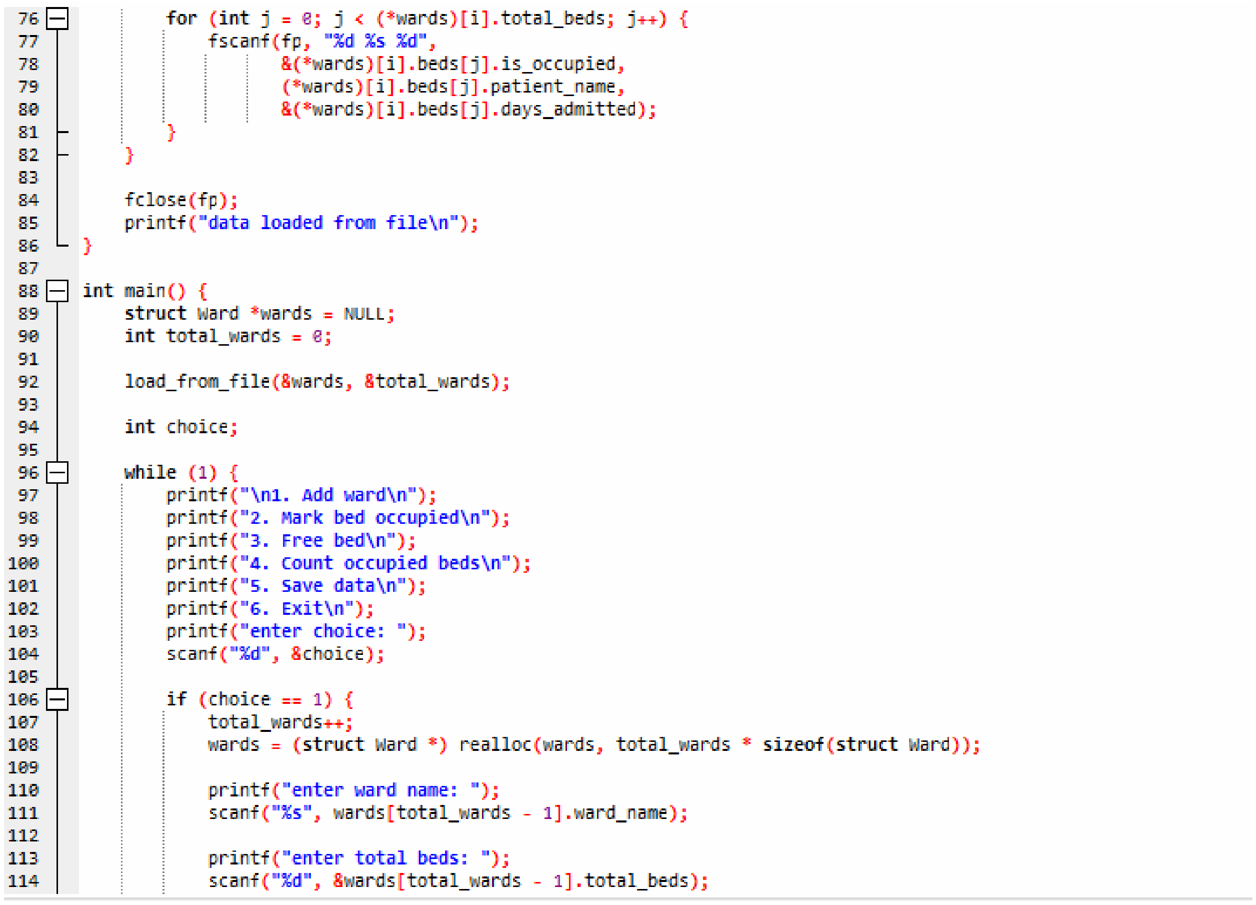
Question #2



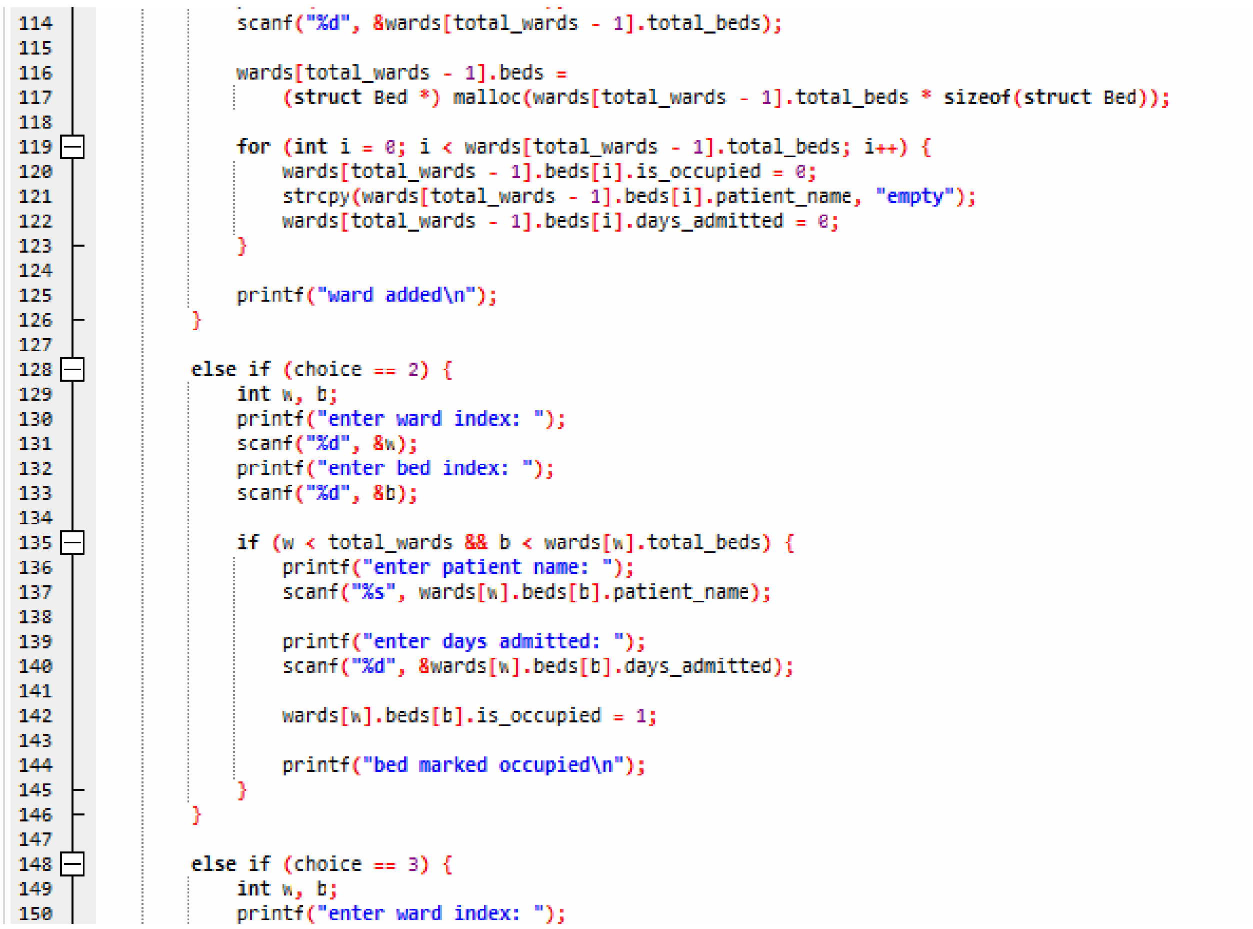
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Bed {
    int is_occupied;
    char patient_name[50];
    int days_admitted;
};

struct Ward {
    char ward_name[50];
    int total_beds;
    struct Bed *beds;
};

int count_occupied_recursive(struct Ward *wards, int ward_index, int bed_index) {
    if (ward_index < 0) {
        return 0;
    }

    if (bed_index < 0) {
        return count_occupied_recursive(wards, ward_index - 1, wards[ward_index - 1].total_beds - 1);
    }

    int count = 0;

    if (wards[ward_index].beds[bed_index].is_occupied == 1) {
        count = 1;
    }

    return count + count_occupied_recursive(wards, ward_index, bed_index - 1);
}

void save_to_file(struct Ward *wards, int total_wards) {
    FILE *fp = fopen("hospital_data.txt", "w");
    if (fp == NULL) {
        printf("file error\n");
        return;
```



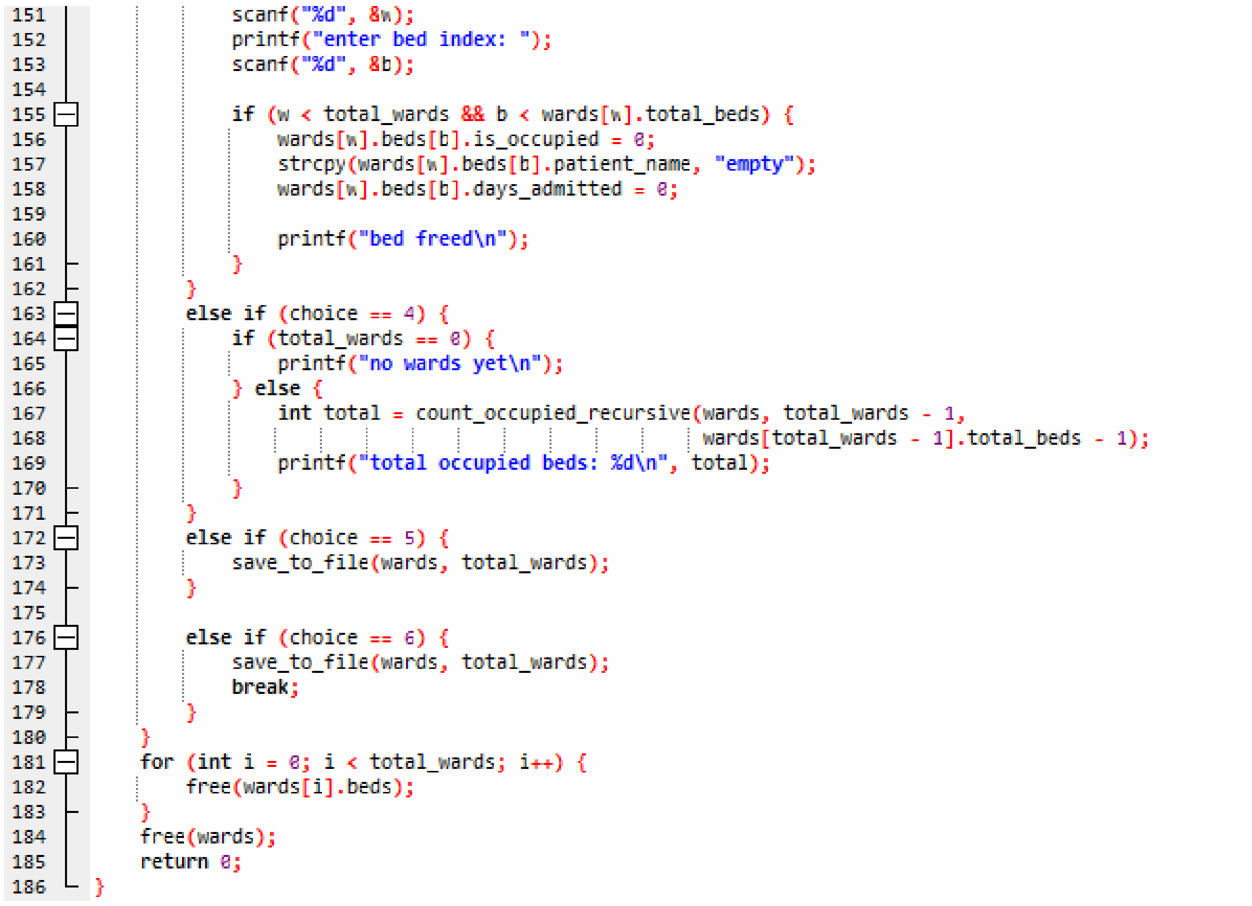
```c
        return;
    }

    fprintf(fp, "%d\n", total_wards);
int i;
    for (int i = 0; i < total_wards; i++) {
        fprintf(fp, "%s %d\n", wards[i].ward_name, wards[i].total_beds);

        for (int j = 0; j < wards[i].total_beds; j++) {
            fprintf(fp, "%d %s %d\n",
                    wards[i].beds[j].is_occupied,
                    wards[i].beds[j].patient_name,
                    wards[i].beds[j].days_admitted);
        }
    }

    fclose(fp);
    printf("data saved to hospital_data.txt\n");
}

void load_from_file(struct Ward **wards, int *total_wards) {
    FILE *fp = fopen("hospital_data.txt", "r");
    if (fp == NULL) {
        printf("file not found, starting fresh...\n");
        return;
    }

    fscanf(fp, "%d", total_wards);

    *wards = (struct Ward *) malloc((*total_wards) * sizeof(struct Ward));

    for (int i = 0; i < *total_wards; i++) {
        fscanf(fp, "%s %d", (*wards)[i].ward_name, &(*wards)[i].total_beds);

        (*wards)[i].beds =
            (struct Bed *) malloc((*wards)[i].total_beds * sizeof(struct Bed));

```



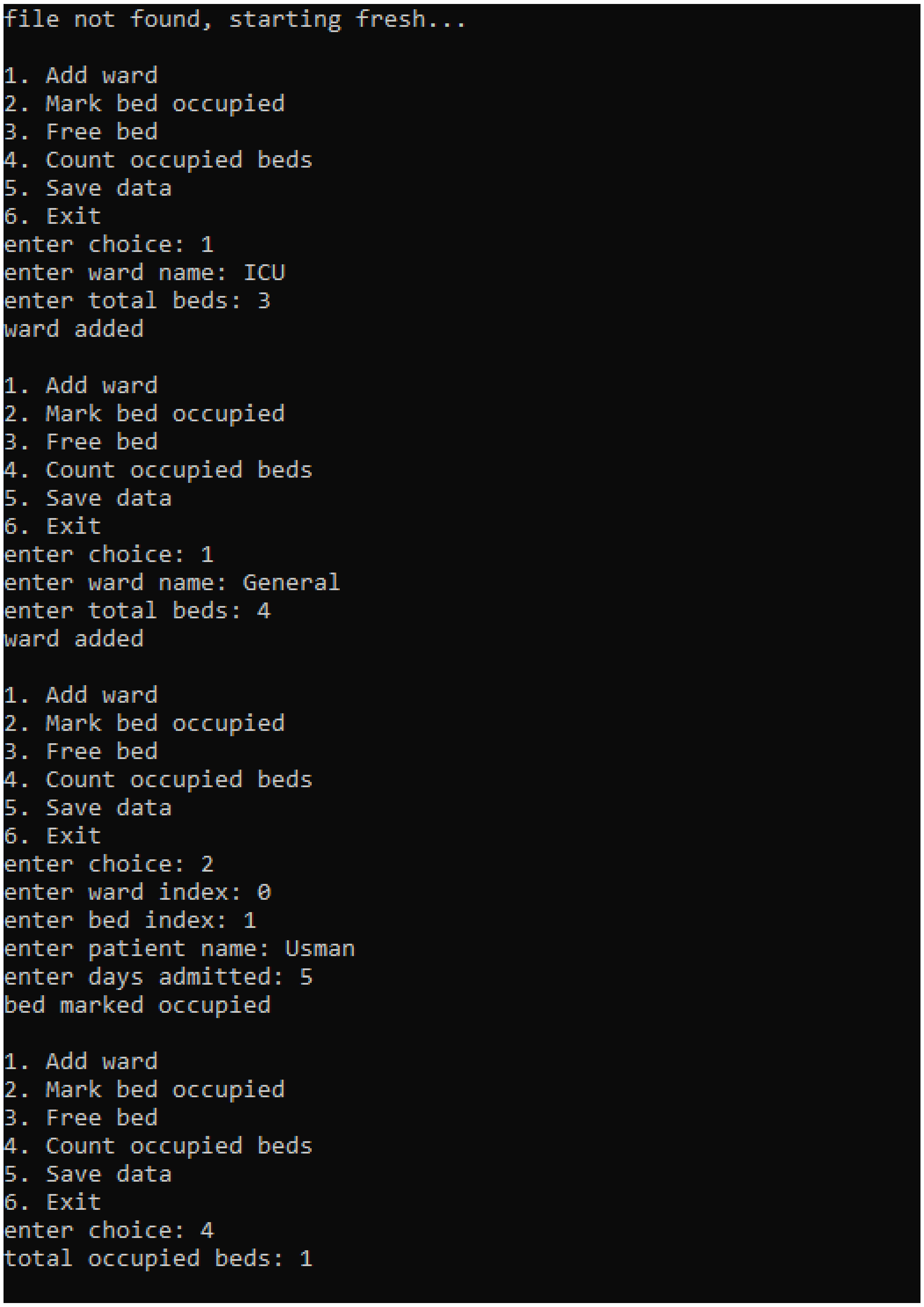
```c
        for (int j = 0; j < (*wards)[i].total_beds; j++) {
            fscanf(fp, "%d %s %d",
                   &(*wards)[i].beds[j].is_occupied,
                   (*wards)[i].beds[j].patient_name,
                   &(*wards)[i].beds[j].days_admitted);
        }
    }

    fclose(fp);
    printf("data loaded from file\n");
}

int main() {
    struct Ward *wards = NULL;
    int total_wards = 0;

    load_from_file(&wards, &total_wards);

    int choice;

    while (1) {
        printf("\n1. Add ward\n");
        printf("2. Mark bed occupied\n");
        printf("3. Free bed\n");
        printf("4. Count occupied beds\n");
        printf("5. Save data\n");
        printf("6. Exit\n");
        printf("enter choice: ");
        scanf("%d", &choice);

        if (choice == 1) {
            total_wards++;
            wards = (struct Ward *) realloc(wards, total_wards * sizeof(struct Ward));

            printf("enter ward name: ");
            scanf("%s", wards[total_wards - 1].ward_name);

            printf("enter total beds: ");
            scanf("%d", &wards[total_wards - 1].total_beds);
```



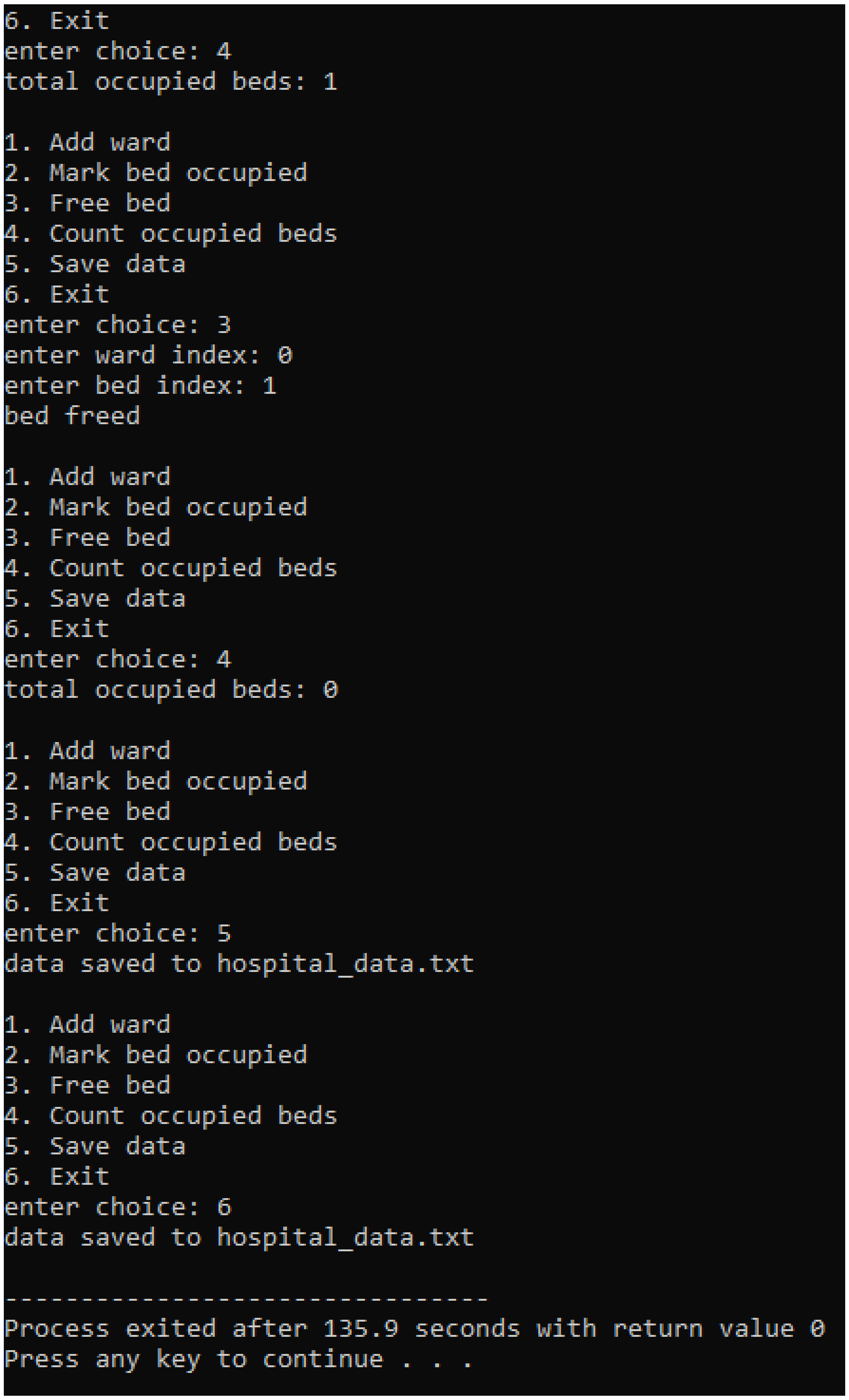
```c
            scanf("%d", &wards[total_wards - 1].total_beds);

            wards[total_wards - 1].beds =
                (struct Bed *) malloc(wards[total_wards - 1].total_beds * sizeof(struct Bed));

            for (int i = 0; i < wards[total_wards - 1].total_beds; i++) {
                wards[total_wards - 1].beds[i].is_occupied = 0;
                strcpy(wards[total_wards - 1].beds[i].patient_name, "empty");
                wards[total_wards - 1].beds[i].days_admitted = 0;
            }

            printf("ward added\n");
        }

        else if (choice == 2) {
            int w, b;
            printf("enter ward index: ");
            scanf("%d", &w);
            printf("enter bed index: ");
            scanf("%d", &b);

            if (w < total_wards && b < wards[w].total_beds) {
                printf("enter patient name: ");
                scanf("%s", wards[w].beds[b].patient_name);

                printf("enter days admitted: ");
                scanf("%d", &wards[w].beds[b].days_admitted);

                wards[w].beds[b].is_occupied = 1;

                printf("bed marked occupied\n");
            }
        }

        else if (choice == 3) {
            int w, b;
            printf("enter ward index: ");
```



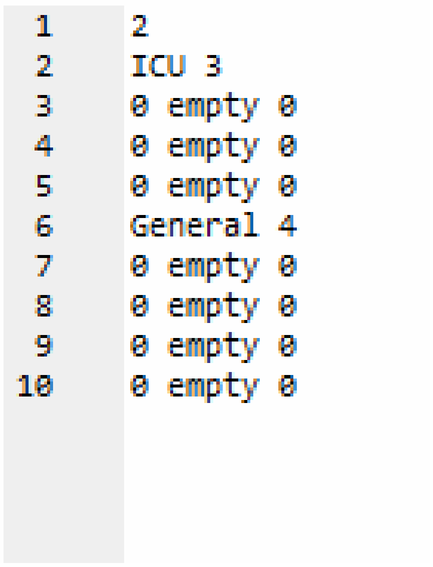
```c
            scanf("%d", &w);
            printf("enter bed index: ");
            scanf("%d", &b);

            if (w < total_wards && b < wards[w].total_beds) {
                wards[w].beds[b].is_occupied = 0;
                strcpy(wards[w].beds[b].patient_name, "empty");
                wards[w].beds[b].days_admitted = 0;

                printf("bed freed\n");
            }
        }
        else if (choice == 4) {
            if (total_wards == 0) {
                printf("no wards yet\n");
            } else {
                int total = count_occupied_recursive(wards, total_wards - 1,
                                                     wards[total_wards - 1].total_beds - 1);
                printf("total occupied beds: %d\n", total);
            }
        }
        else if (choice == 5) {
            save_to_file(wards, total_wards);
        }

        else if (choice == 6) {
            save_to_file(wards, total_wards);
            break;
        }
    }
    for (int i = 0; i < total_wards; i++) {
        free(wards[i].beds);
    }
    free(wards);
    return 0;
}
```



```
file not found, starting fresh...

1. Add ward
2. Mark bed occupied
3. Free bed
4. Count occupied beds
5. Save data
6. Exit
enter choice: 1
enter ward name: ICU
enter total beds: 3
ward added

1. Add ward
2. Mark bed occupied
3. Free bed
4. Count occupied beds
5. Save data
6. Exit
enter choice: 1
enter ward name: General
enter total beds: 4
ward added

1. Add ward
2. Mark bed occupied
3. Free bed
4. Count occupied beds
5. Save data
6. Exit
enter choice: 2
enter ward index: 0
enter bed index: 1
enter patient name: Usman
enter days admitted: 5
bed marked occupied

1. Add ward
2. Mark bed occupied
3. Free bed
4. Count occupied beds
5. Save data
6. Exit
enter choice: 4
total occupied beds: 1
```

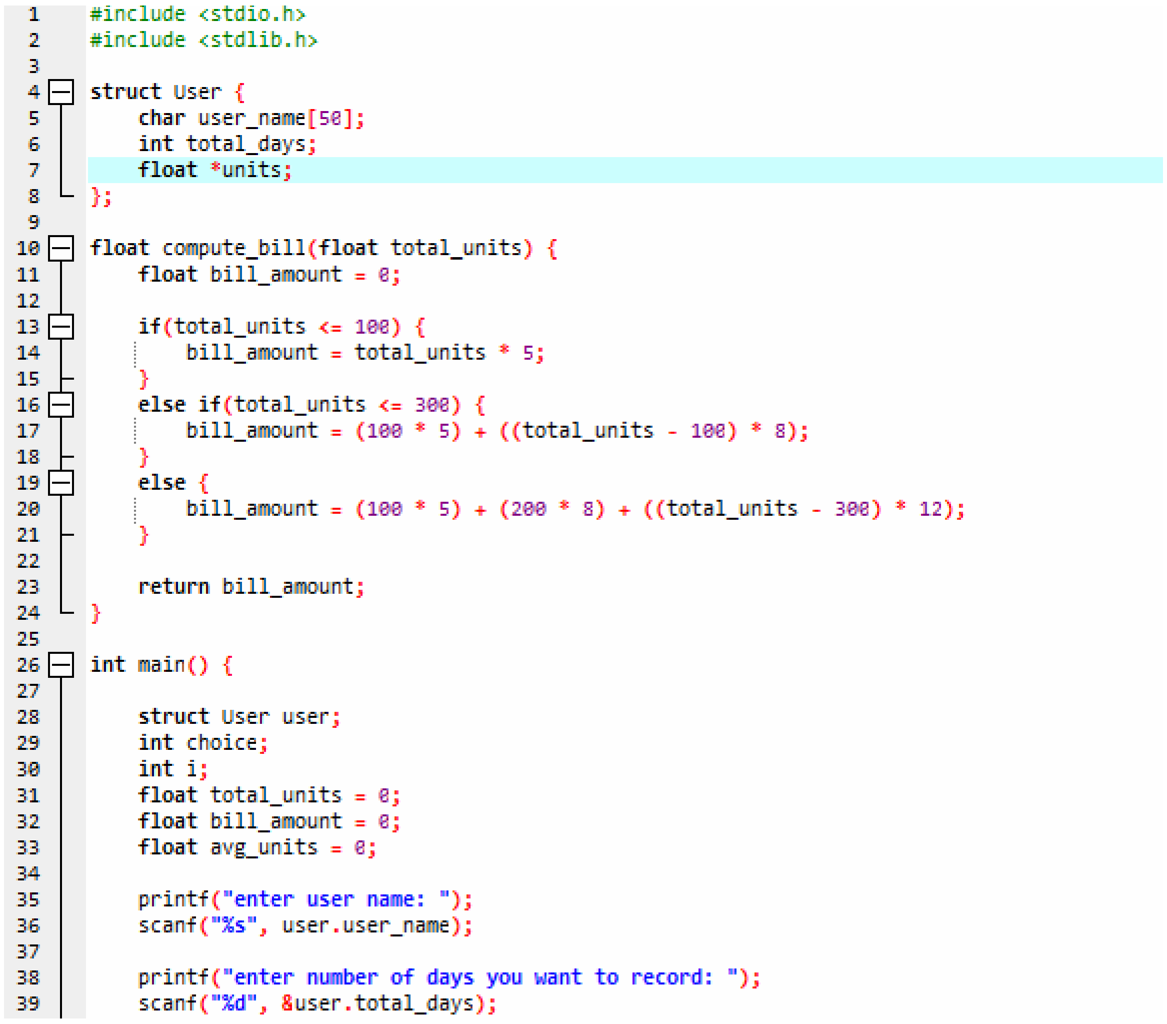


```
6. Exit
enter choice: 4
total occupied beds: 1

1. Add ward
2. Mark bed occupied
3. Free bed
4. Count occupied beds
5. Save data
6. Exit
enter choice: 3
enter ward index: 0
enter bed index: 1
bed freed

1. Add ward
2. Mark bed occupied
3. Free bed
4. Count occupied beds
5. Save data
6. Exit
enter choice: 4
total occupied beds: 0

1. Add ward
2. Mark bed occupied
3. Free bed
4. Count occupied beds
5. Save data
6. Exit
enter choice: 5
data saved to hospital_data.txt

1. Add ward
2. Mark bed occupied
3. Free bed
4. Count occupied beds
5. Save data
6. Exit
enter choice: 6
data saved to hospital_data.txt

--------------------------------
Process exited after 135.9 seconds with return value 0
Press any key to continue . . .
```

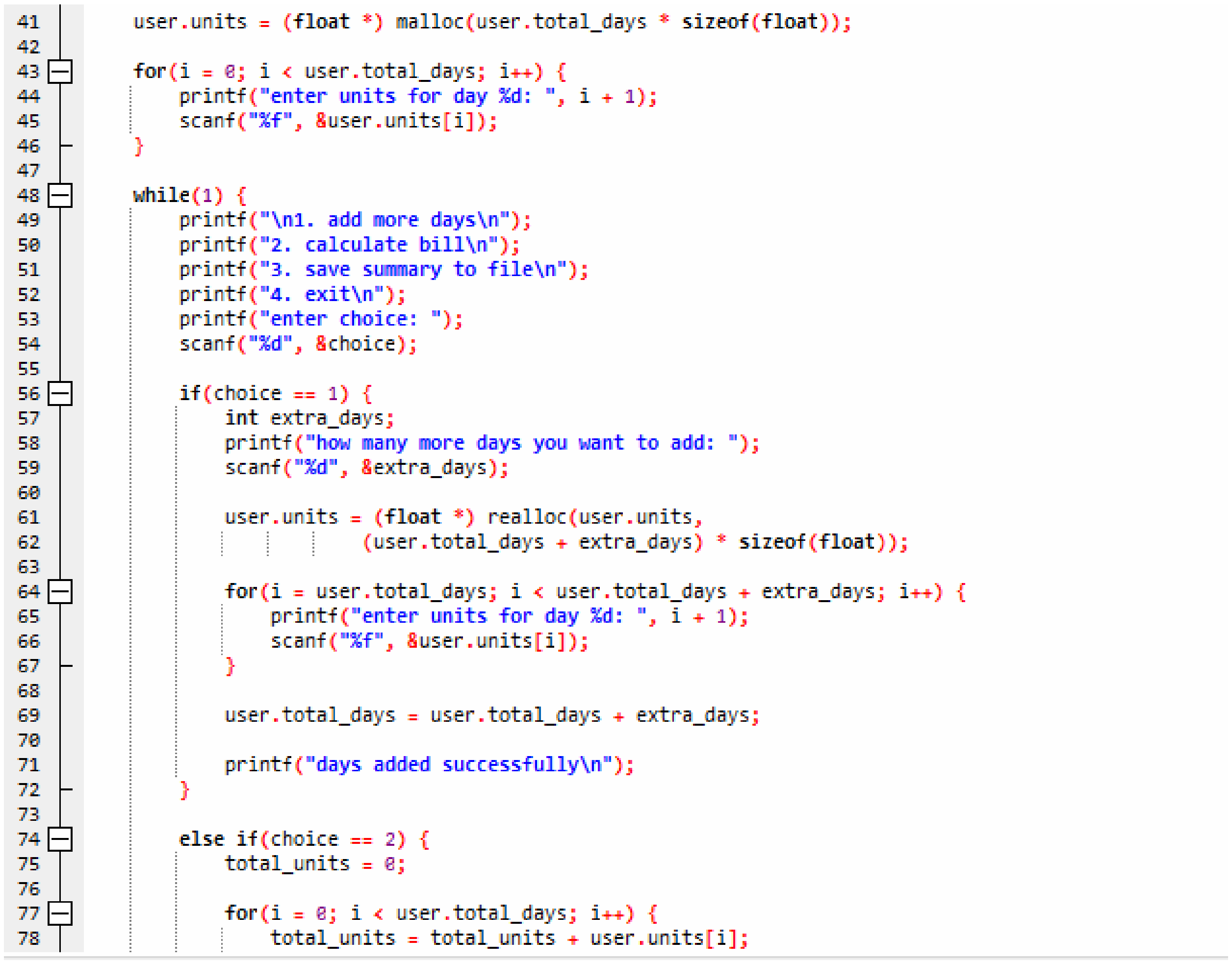


```
2
ICU 3
0 empty 0
0 empty 0
0 empty 0
General 4
0 empty 0
0 empty 0
0 empty 0
0 empty 0
```

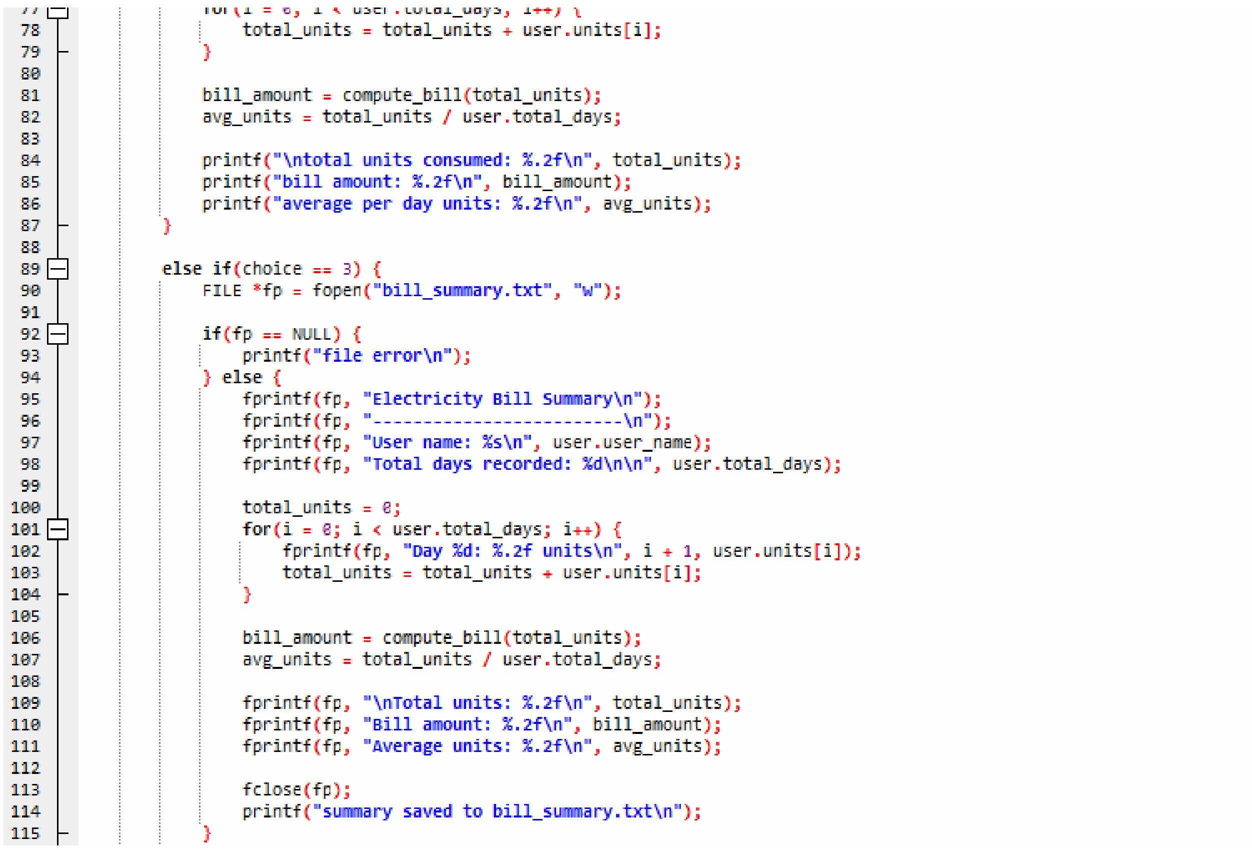
Question #3



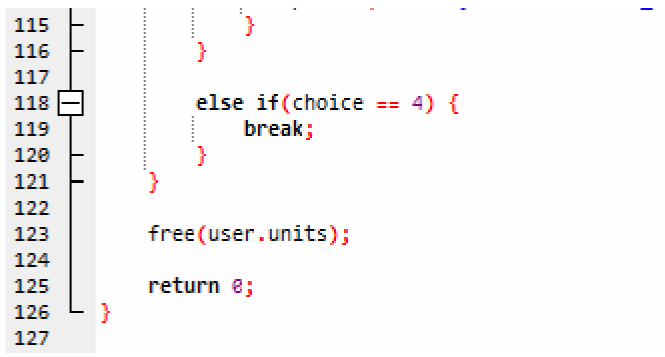
```c
#include <stdio.h>
#include <stdlib.h>

struct User {
    char user_name[50];
    int total_days;
    float *units;
};

float compute_bill(float total_units) {
    float bill_amount = 0;

    if(total_units <= 100) {
        bill_amount = total_units * 5;
    }
    else if(total_units <= 300) {
        bill_amount = (100 * 5) + ((total_units - 100) * 8);
    }
    else {
        bill_amount = (100 * 5) + (200 * 8) + ((total_units - 300) * 12);
    }

    return bill_amount;
}

int main() {

    struct User user;
    int choice;
    int i;
    float total_units = 0;
    float bill_amount = 0;
    float avg_units = 0;

    printf("enter user name: ");
    scanf("%s", user.user_name);

    printf("enter number of days you want to record: ");
    scanf("%d", &user.total_days);
```



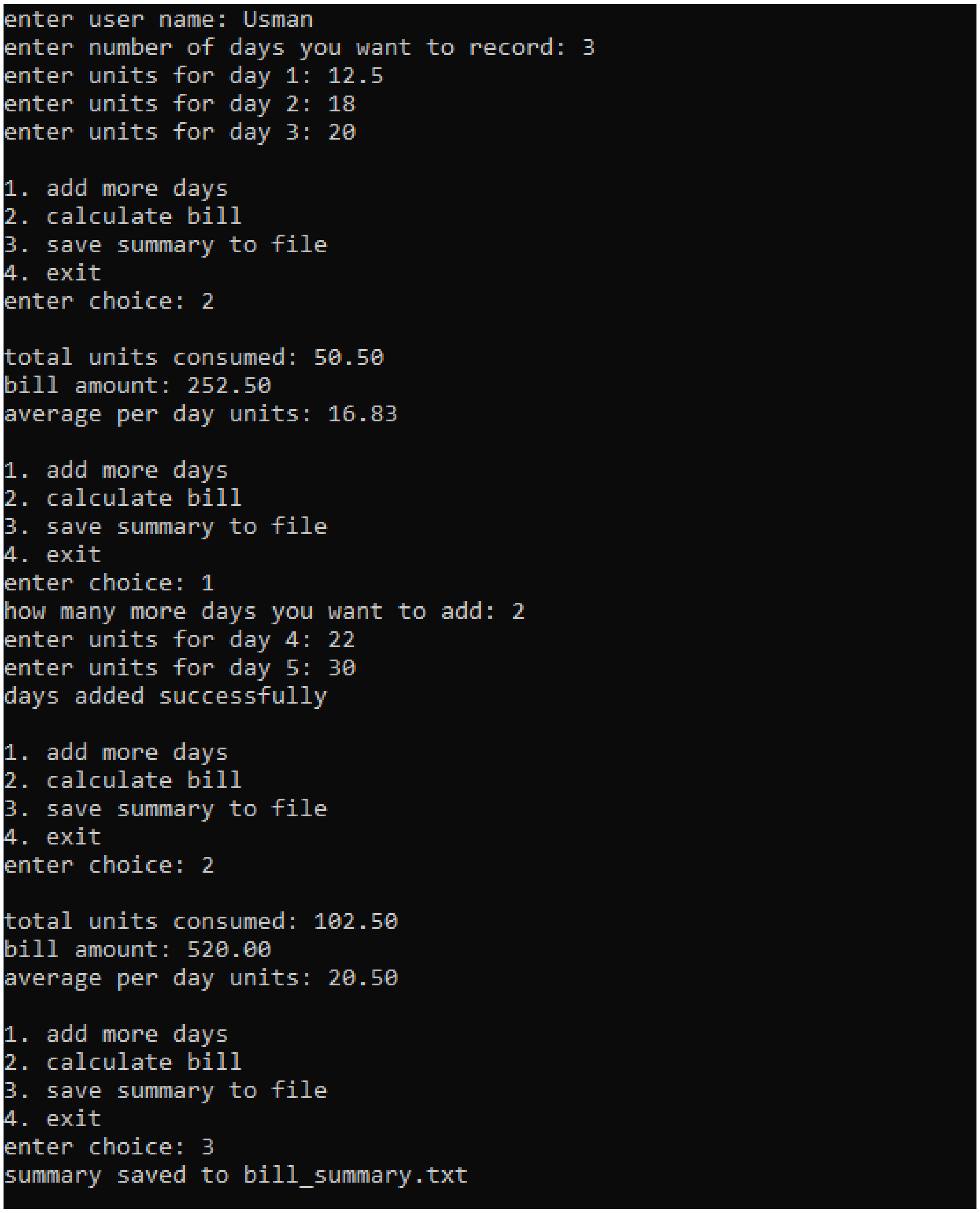
```c
    user.units = (float *) malloc(user.total_days * sizeof(float));

    for(i = 0; i < user.total_days; i++) {
        printf("enter units for day %d: ", i + 1);
        scanf("%f", &user.units[i]);
    }

    while(1) {
        printf("\n1. add more days\n");
        printf("2. calculate bill\n");
        printf("3. save summary to file\n");
        printf("4. exit\n");
        printf("enter choice: ");
        scanf("%d", &choice);

        if(choice == 1) {
            int extra_days;
            printf("how many more days you want to add: ");
            scanf("%d", &extra_days);

            user.units = (float *) realloc(user.units,
                            (user.total_days + extra_days) * sizeof(float));

            for(i = user.total_days; i < user.total_days + extra_days; i++) {
                printf("enter units for day %d: ", i + 1);
                scanf("%f", &user.units[i]);
            }

            user.total_days = user.total_days + extra_days;

            printf("days added successfully\n");
        }

        else if(choice == 2) {
            total_units = 0;

            for(i = 0; i < user.total_days; i++) {
                total_units = total_units + user.units[i];
```



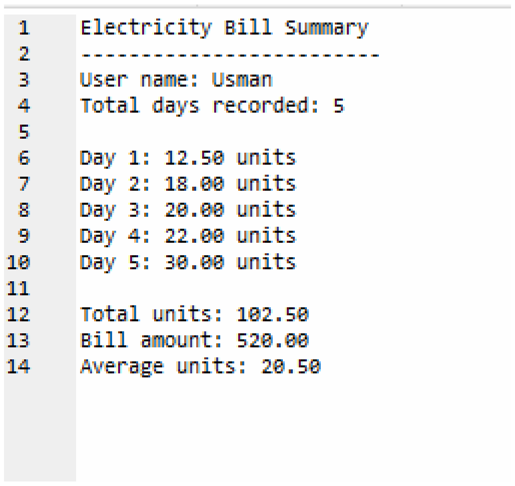
```c
            for(i = 0; i < user.total_days; i++) {
                total_units = total_units + user.units[i];
            }

            bill_amount = compute_bill(total_units);
            avg_units = total_units / user.total_days;

            printf("\ntotal units consumed: %.2f\n", total_units);
            printf("bill amount: %.2f\n", bill_amount);
            printf("average per day units: %.2f\n", avg_units);
        }

        else if(choice == 3) {
            FILE *fp = fopen("bill_summary.txt", "w");

            if(fp == NULL) {
                printf("file error\n");
            } else {
                fprintf(fp, "Electricity Bill Summary\n");
                fprintf(fp, "-------------------------\n");
                fprintf(fp, "User name: %s\n", user.user_name);
                fprintf(fp, "Total days recorded: %d\n\n", user.total_days);

                total_units = 0;
                for(i = 0; i < user.total_days; i++) {
                    fprintf(fp, "Day %d: %.2f units\n", i + 1, user.units[i]);
                    total_units = total_units + user.units[i];
                }

                bill_amount = compute_bill(total_units);
                avg_units = total_units / user.total_days;

                fprintf(fp, "\nTotal units: %.2f\n", total_units);
                fprintf(fp, "Bill amount: %.2f\n", bill_amount);
                fprintf(fp, "Average units: %.2f\n", avg_units);

                fclose(fp);
                printf("summary saved to bill_summary.txt\n");
            }
```



```c
            }
        }

        else if(choice == 4) {
            break;
        }
    }

    free(user.units);

    return 0;
}
```



```
enter user name: Usman
enter number of days you want to record: 3
enter units for day 1: 12.5
enter units for day 2: 18
enter units for day 3: 20

1. add more days
2. calculate bill
3. save summary to file
4. exit
enter choice: 2

total units consumed: 50.50
bill amount: 252.50
average per day units: 16.83

1. add more days
2. calculate bill
3. save summary to file
4. exit
enter choice: 1
how many more days you want to add: 2
enter units for day 4: 22
enter units for day 5: 30
days added successfully

1. add more days
2. calculate bill
3. save summary to file
4. exit
enter choice: 2

total units consumed: 102.50
bill amount: 520.00
average per day units: 20.50

1. add more days
2. calculate bill
3. save summary to file
4. exit
enter choice: 3
summary saved to bill_summary.txt
```



```
Electricity Bill Summary
-------------------------
User name: Usman
Total days recorded: 5

Day 1: 12.50 units
Day 2: 18.00 units
Day 3: 20.00 units
Day 4: 22.00 units
Day 5: 30.00 units

Total units: 102.50
Bill amount: 520.00
Average units: 20.50
```