UCS 2201 Fundamentals and Practice of Software Development

A4: Programs using Arrays

Batch 2023-2027
Name: SK Shivaanee
Register ID: 2310257
Section: CSE B
Dr. Chitra Babu, Dr. D. Thenmozhi, Dr. M. Saritha

Learning Outcome:

You will be able to implement functions in C with the following features:

● Using one-dimensional and two-dimensional arrays

● Passing arrays to a function

Best Practices:

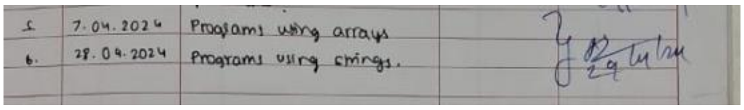
You will be able to adapt to the following best practices

● Modular programming

● To develop code incrementally

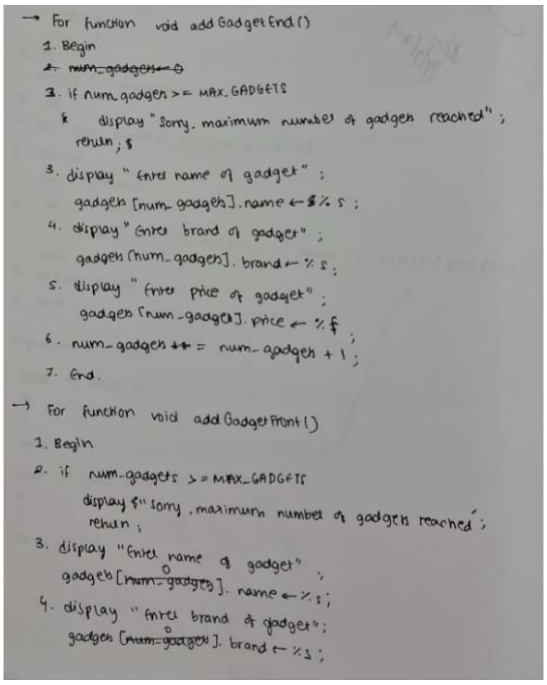
● Pass arrays to function

Assignment: Solve the following problems by implementing in C using arrays. (CO7, K3, 1.3.1, 1.4.1, 2.1.2, 2.1.3, 4.2.1, 14.2.1, 14.2.2). a) Write pseudo code for all the functions. b) Develop a program to test the functions with at least three test cases.

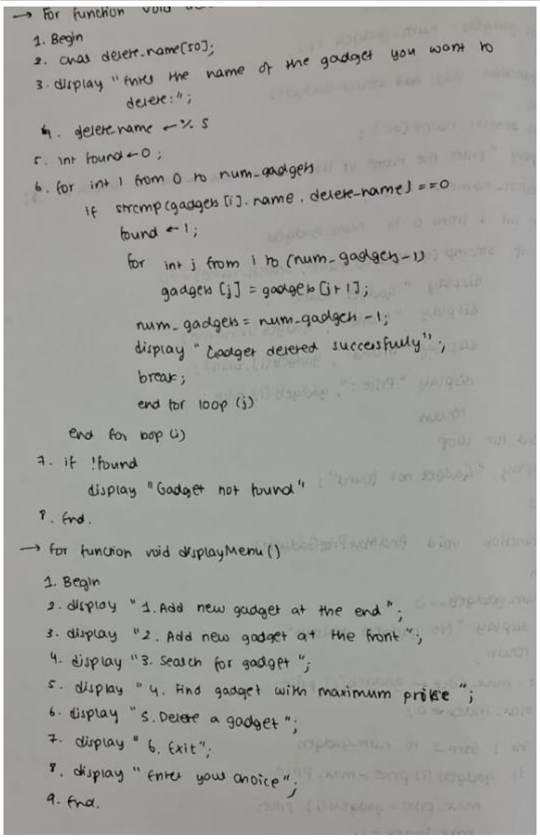


5	7.04.2024	Programs using arrays		[illegible]
6.	28.04.2024	Programs using strings.		

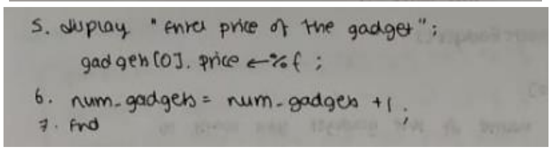
1. The information about gadgets namely Product, Brand and Price are maintained in three different arrays. Assume there are 5 gadget details available initially. Write a menu driven program to do the following i. Add new gadget details in to the arrays at the end position ii. Add new gadget details in to the arrays at the front position iii. Search for information about any given gadget. iv. Find the gadget with the maximum price. v. Delete a given gadget from the arrays.



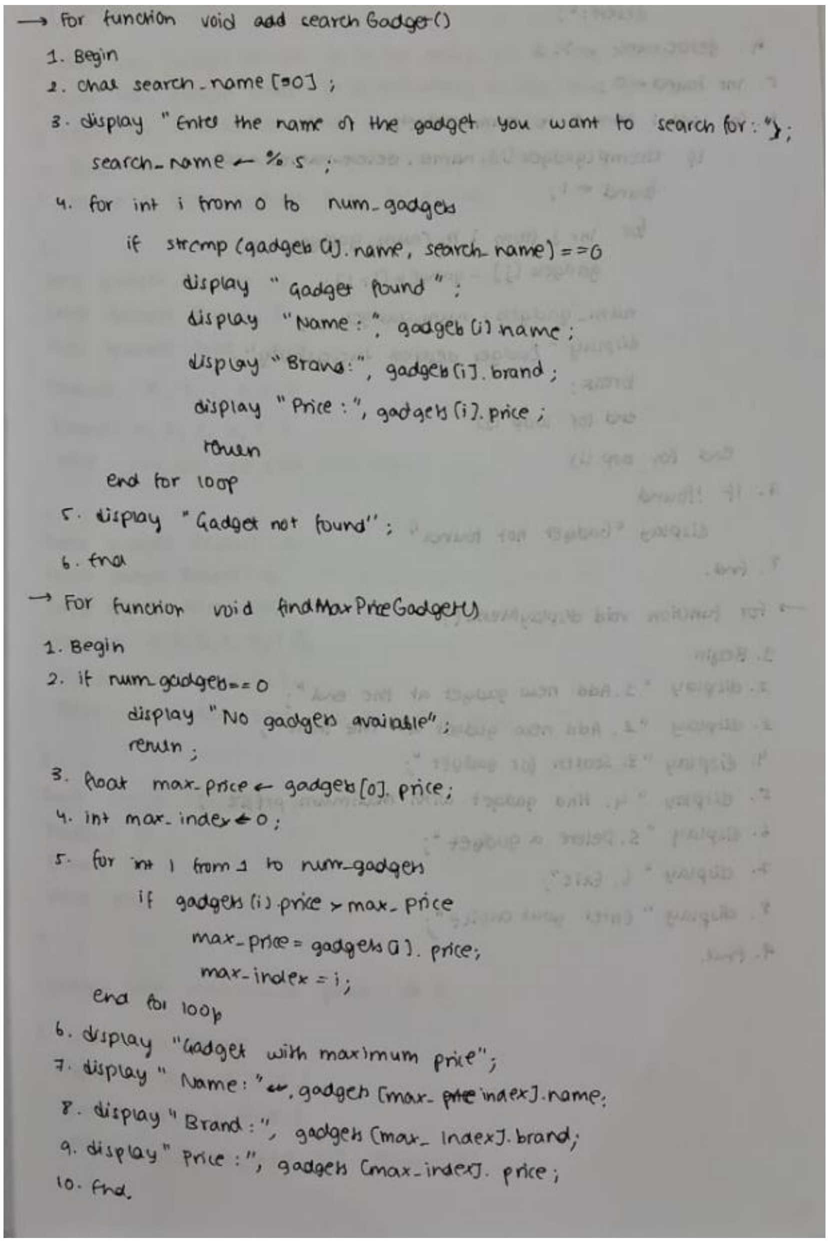
```
→ For function void addGadgetEnd()
  1. Begin
  ~~2. num_gadgets ← 0~~
  3. if num_gadgets >= MAX_GADGETS
     { display "Sorry. maximum number of gadgets reached";
       return; }
  3. display "Enter name of gadget";
     gadgets[num_gadgets].name ← ~~$~~ %s;
  4. display "Enter brand of gadget";
     gadgets[num_gadgets].brand ← %s;
  5. display "Enter price of gadget";
     gadgets[num_gadgets].price ← %f;
  6. num_gadgets ~~++~~ = num_gadgets + 1;
  7. End.

→ For function void addGadgetFront()
  1. Begin
  2. if num_gadgets >= MAX_GADGETS
        display ~~$~~ "Sorry, maximum number of gadgets reached";
        return;
  3. display "Enter name of gadget";
     gadgets[~~num_gadgets~~ 0].name ← %s;
  4. display "Enter brand of gadget";
     gadgets[~~num_gadgets~~ 0].brand ← %s;
```



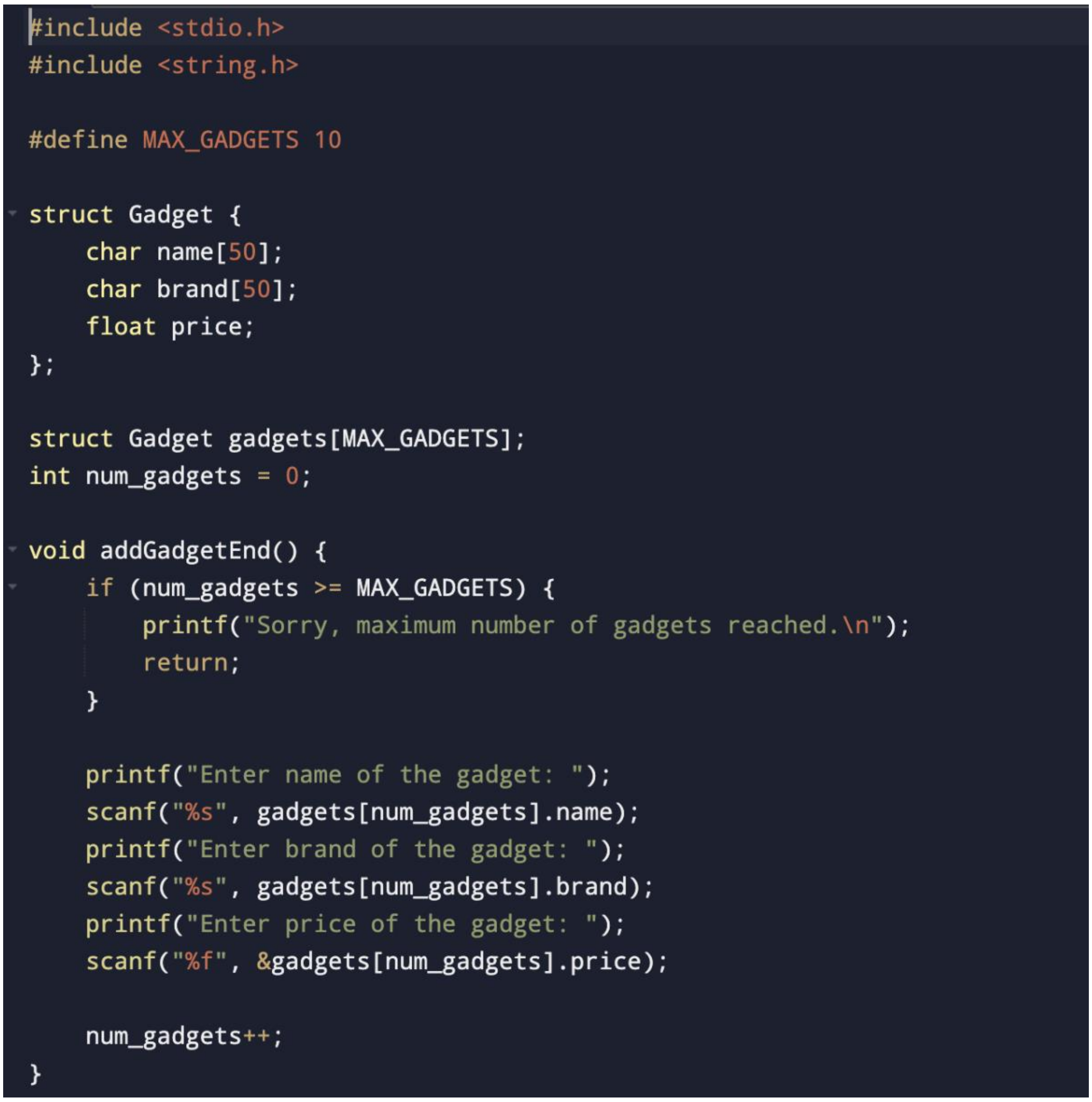
```
→ For function void
  1. Begin
  2. char delete_name[50];
  3. display "Enter the name of the gadget you want to
            delete:";
  4. delete_name ← %s
  5. int found ← 0;
  6. for int i from 0 to num_gadgets
        if strcmp(gadgets[i].name, delete_name) == 0
           found ← 1;
            for int j from i to (num_gadgets - 1)
                gadgets[j] = gadgets[j+1];
            num_gadgets = num_gadgets - 1;
            display "Gadget deleted successfully";
            break;
            end for loop (j)
     end for loop (i)
  7. if !found
        display "Gadget not found"
  8. End.

→ for function void displayMenu()
  1. Begin
  2. display "1. Add new gadget at the end";
  3. display "2. Add new gadget at the front";
  4. display "3. Search for gadget";
  5. display "4. Find gadget with maximum price";
  6. display "5. Delete a gadget";
  7. display "6. Exit";
  8. display "Enter your choice";
  9. End.
```



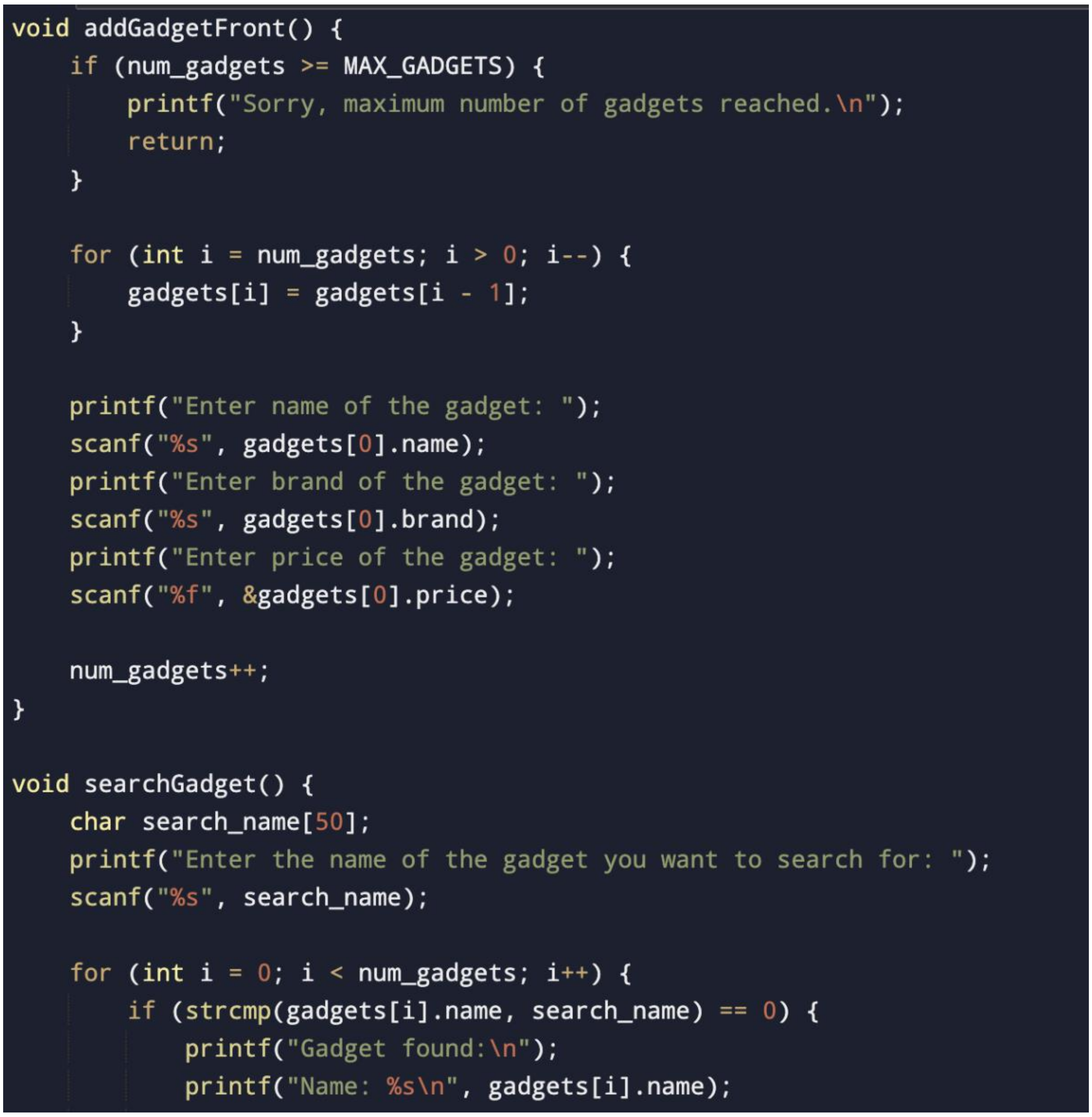
```
5. display "Enter price of the gadget";
     gadgets[0].price ← %f;
6. num_gadgets = num_gadgets + 1;
7. End
```



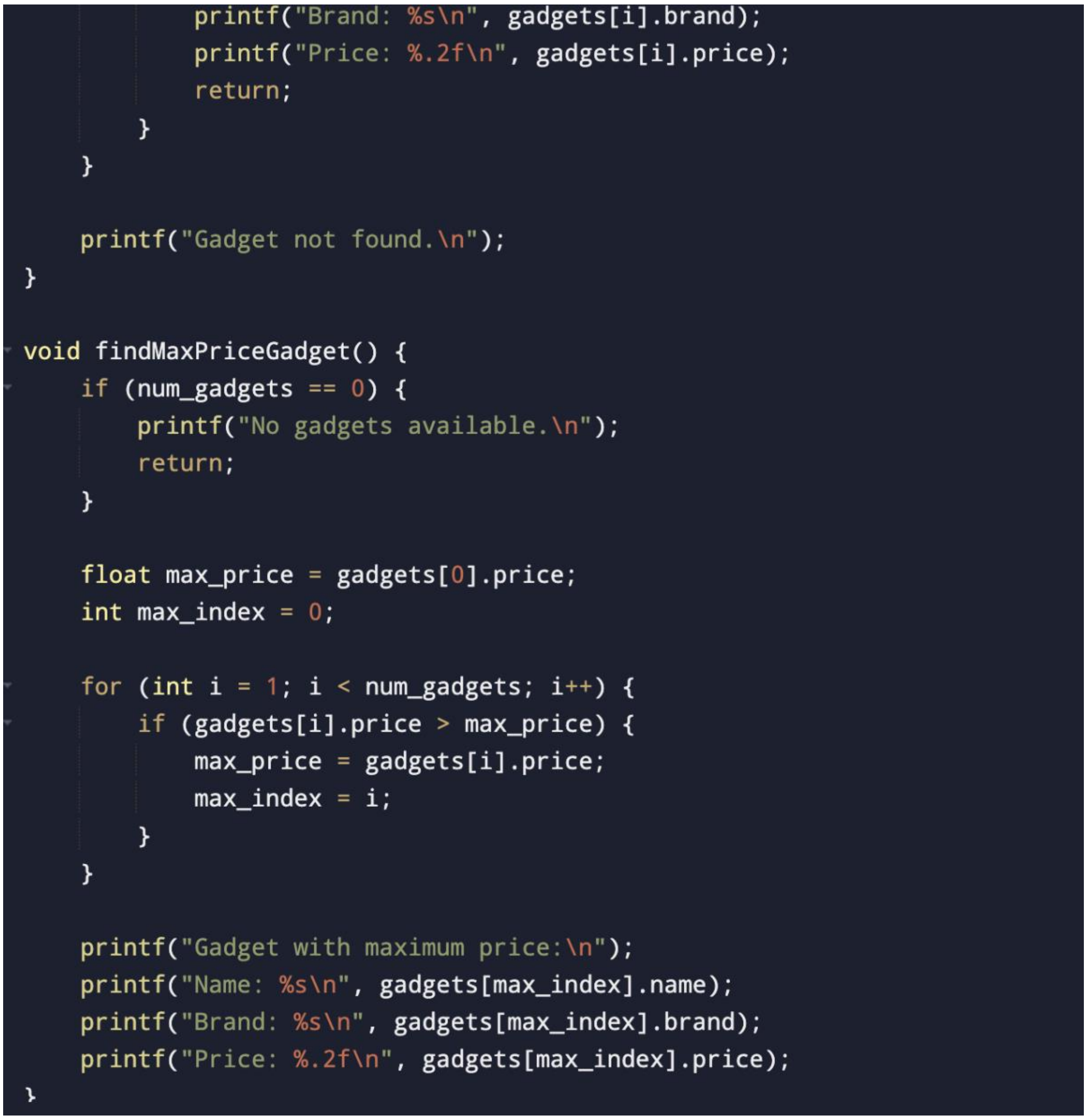
```
→ For function void ~~add~~ search Gadget()
  1. Begin
  2. char search_name [50] ;
  3. display "Enter the name of the gadget you want to search for:";
     search_name ← %s ;
  4. for int i from 0 to num_gadgets
          if strcmp (gadgets [i].name, search_name) ==0
               display " Gadget found " ;
               display "Name : ", gadgets [i].name ;
               display "Brand:", gadgets [i]. brand ;
               display "Price :", gadgets [i]. price ;
               return
      end for loop
  5. display " Gadget not found" ;
  6. End
→ For function void findMaxPriceGadget()
  1. Begin
  2. if num_gadgets== 0
          display "No gadgets available" ;
          return ;
  3. float max_price ← gadgets [0]. price;
  4. int max_index ← 0 ;
  5. for int i from 1 to num_gadgets
          if gadgets [i].price > max_price
               max_price = gadgets [i]. price;
               max_index = i ;
     end for loop
  6. display "Gadget with maximum price";
  7. display " Name: ", gadgets [max_~~price~~ index].name;
  8. display " Brand :", gadgets [max_ Index]. brand;
  9. display " Price :", gadgets [max_index]. price ;
  10. End.
```



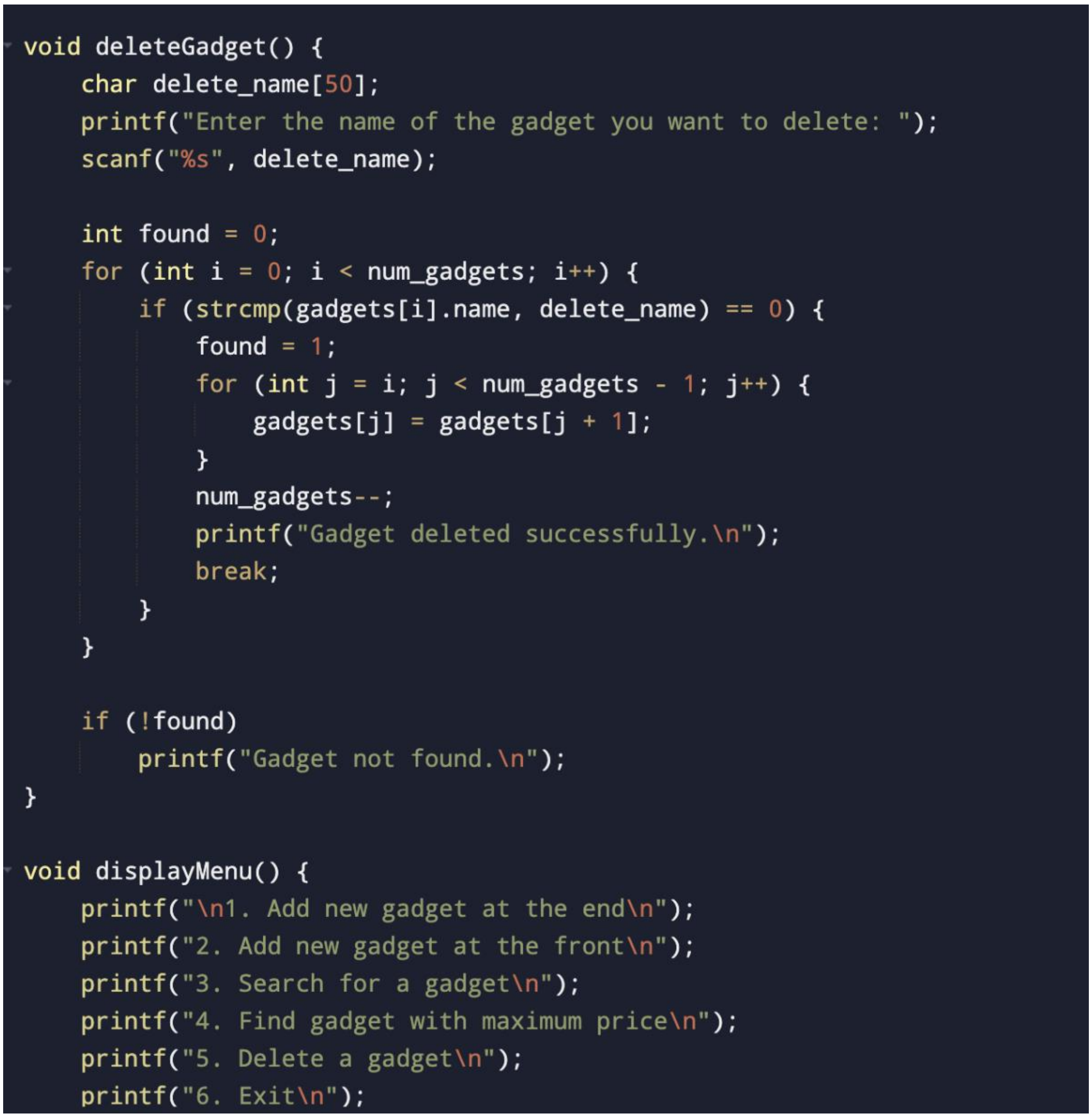
```c
#include <stdio.h>
#include <string.h>

#define MAX_GADGETS 10

struct Gadget {
    char name[50];
    char brand[50];
    float price;
};

struct Gadget gadgets[MAX_GADGETS];
int num_gadgets = 0;

void addGadgetEnd() {
    if (num_gadgets >= MAX_GADGETS) {
        printf("Sorry, maximum number of gadgets reached.\n");
        return;
    }

    printf("Enter name of the gadget: ");
    scanf("%s", gadgets[num_gadgets].name);
    printf("Enter brand of the gadget: ");
    scanf("%s", gadgets[num_gadgets].brand);
    printf("Enter price of the gadget: ");
    scanf("%f", &gadgets[num_gadgets].price);

    num_gadgets++;
}
```



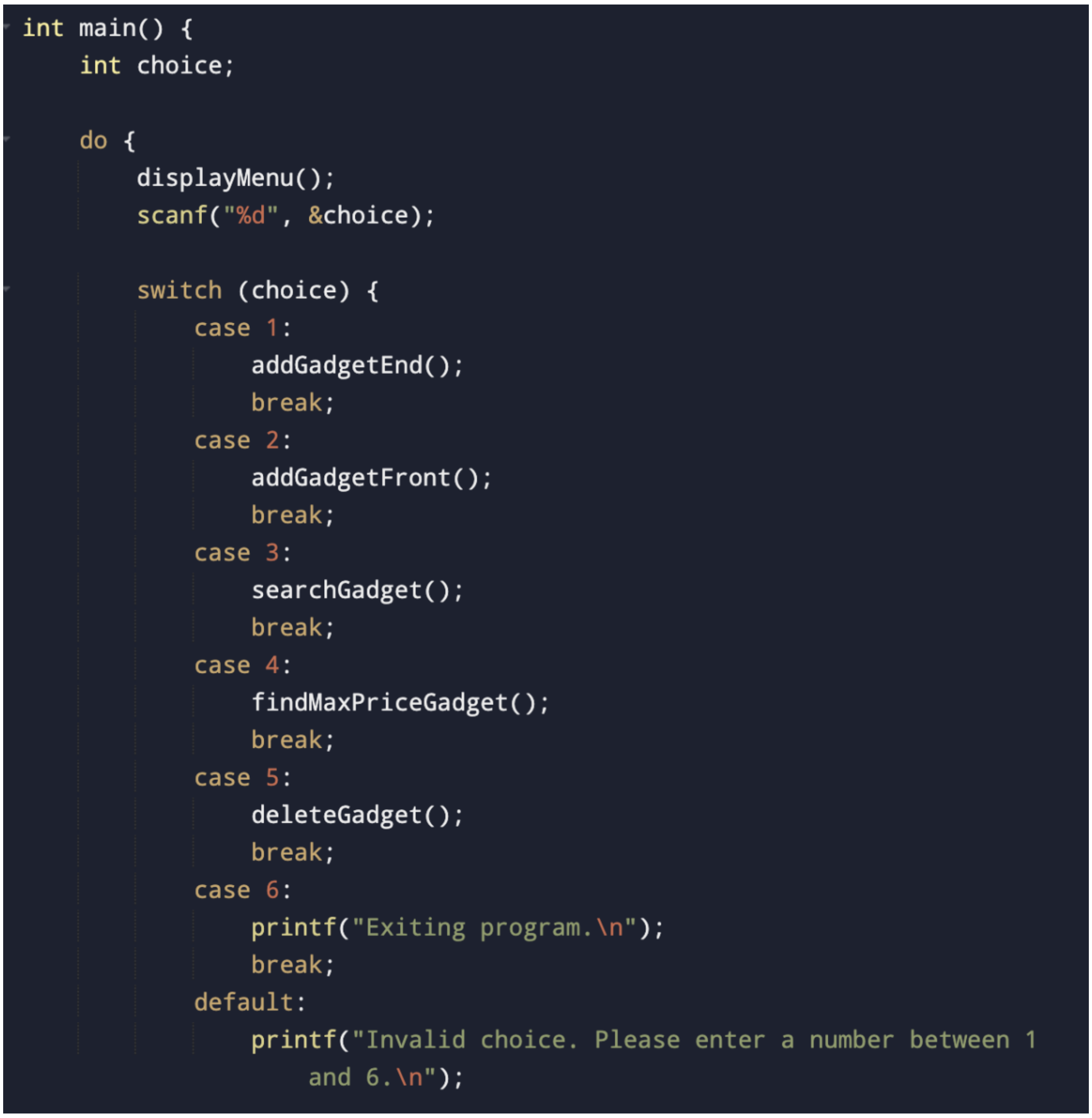
```
void addGadgetFront() {
    if (num_gadgets >= MAX_GADGETS) {
        printf("Sorry, maximum number of gadgets reached.\n");
        return;
    }

    for (int i = num_gadgets; i > 0; i--) {
        gadgets[i] = gadgets[i - 1];
    }

    printf("Enter name of the gadget: ");
    scanf("%s", gadgets[0].name);
    printf("Enter brand of the gadget: ");
    scanf("%s", gadgets[0].brand);
    printf("Enter price of the gadget: ");
    scanf("%f", &gadgets[0].price);

    num_gadgets++;
}

void searchGadget() {
    char search_name[50];
    printf("Enter the name of the gadget you want to search for: ");
    scanf("%s", search_name);

    for (int i = 0; i < num_gadgets; i++) {
        if (strcmp(gadgets[i].name, search_name) == 0) {
            printf("Gadget found:\n");
            printf("Name: %s\n", gadgets[i].name);
```



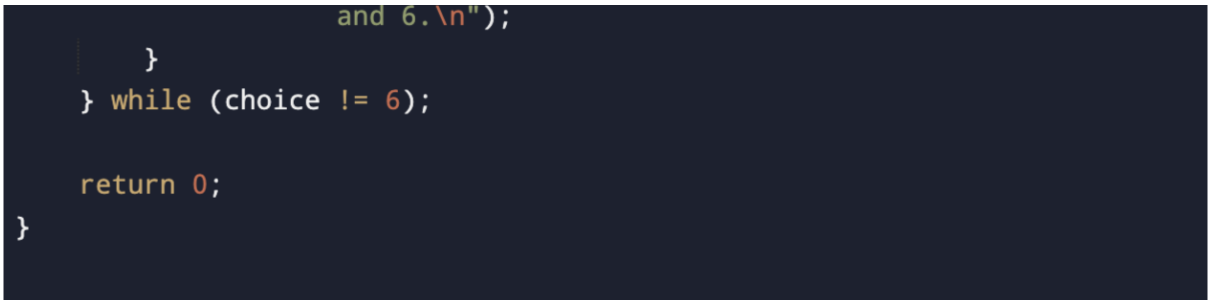
```
            printf("Brand: %s\n", gadgets[i].brand);
            printf("Price: %.2f\n", gadgets[i].price);
            return;
        }
    }

    printf("Gadget not found.\n");
}

void findMaxPriceGadget() {
    if (num_gadgets == 0) {
        printf("No gadgets available.\n");
        return;
    }

    float max_price = gadgets[0].price;
    int max_index = 0;

    for (int i = 1; i < num_gadgets; i++) {
        if (gadgets[i].price > max_price) {
            max_price = gadgets[i].price;
            max_index = i;
        }
    }

    printf("Gadget with maximum price:\n");
    printf("Name: %s\n", gadgets[max_index].name);
    printf("Brand: %s\n", gadgets[max_index].brand);
    printf("Price: %.2f\n", gadgets[max_index].price);
}
```



```c
void deleteGadget() {
    char delete_name[50];
    printf("Enter the name of the gadget you want to delete: ");
    scanf("%s", delete_name);

    int found = 0;
    for (int i = 0; i < num_gadgets; i++) {
        if (strcmp(gadgets[i].name, delete_name) == 0) {
            found = 1;
            for (int j = i; j < num_gadgets - 1; j++) {
                gadgets[j] = gadgets[j + 1];
            }
            num_gadgets--;
            printf("Gadget deleted successfully.\n");
            break;
        }
    }

    if (!found)
        printf("Gadget not found.\n");
}

void displayMenu() {
    printf("\n1. Add new gadget at the end\n");
    printf("2. Add new gadget at the front\n");
    printf("3. Search for a gadget\n");
    printf("4. Find gadget with maximum price\n");
    printf("5. Delete a gadget\n");
    printf("6. Exit\n");
```

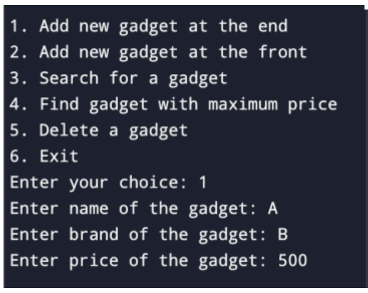


```c
int main() {
    int choice;

    do {
        displayMenu();
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addGadgetEnd();
                break;
            case 2:
                addGadgetFront();
                break;
            case 3:
                searchGadget();
                break;
            case 4:
                findMaxPriceGadget();
                break;
            case 5:
                deleteGadget();
                break;
            case 6:
                printf("Exiting program.\n");
                break;
            default:
                printf("Invalid choice. Please enter a number between 1
                    and 6.\n");
```

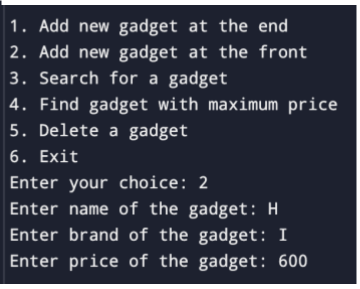


and 6.\n");

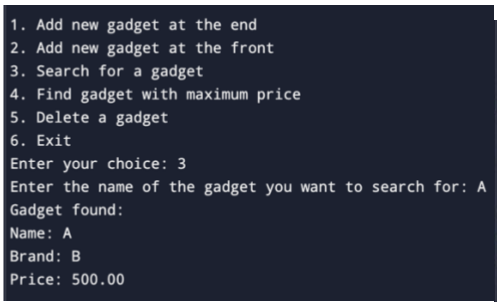
```c
        }
    } while (choice != 6);

    return 0;
}
```



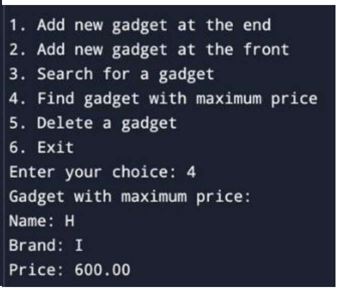
```
1. Add new gadget at the end
2. Add new gadget at the front
3. Search for a gadget
4. Find gadget with maximum price
5. Delete a gadget
6. Exit
Enter your choice: 1
Enter name of the gadget: A
Enter brand of the gadget: B
Enter price of the gadget: 500
```



```
1. Add new gadget at the end
2. Add new gadget at the front
3. Search for a gadget
4. Find gadget with maximum price
5. Delete a gadget
6. Exit
Enter your choice: 2
Enter name of the gadget: H
Enter brand of the gadget: I
Enter price of the gadget: 600
```

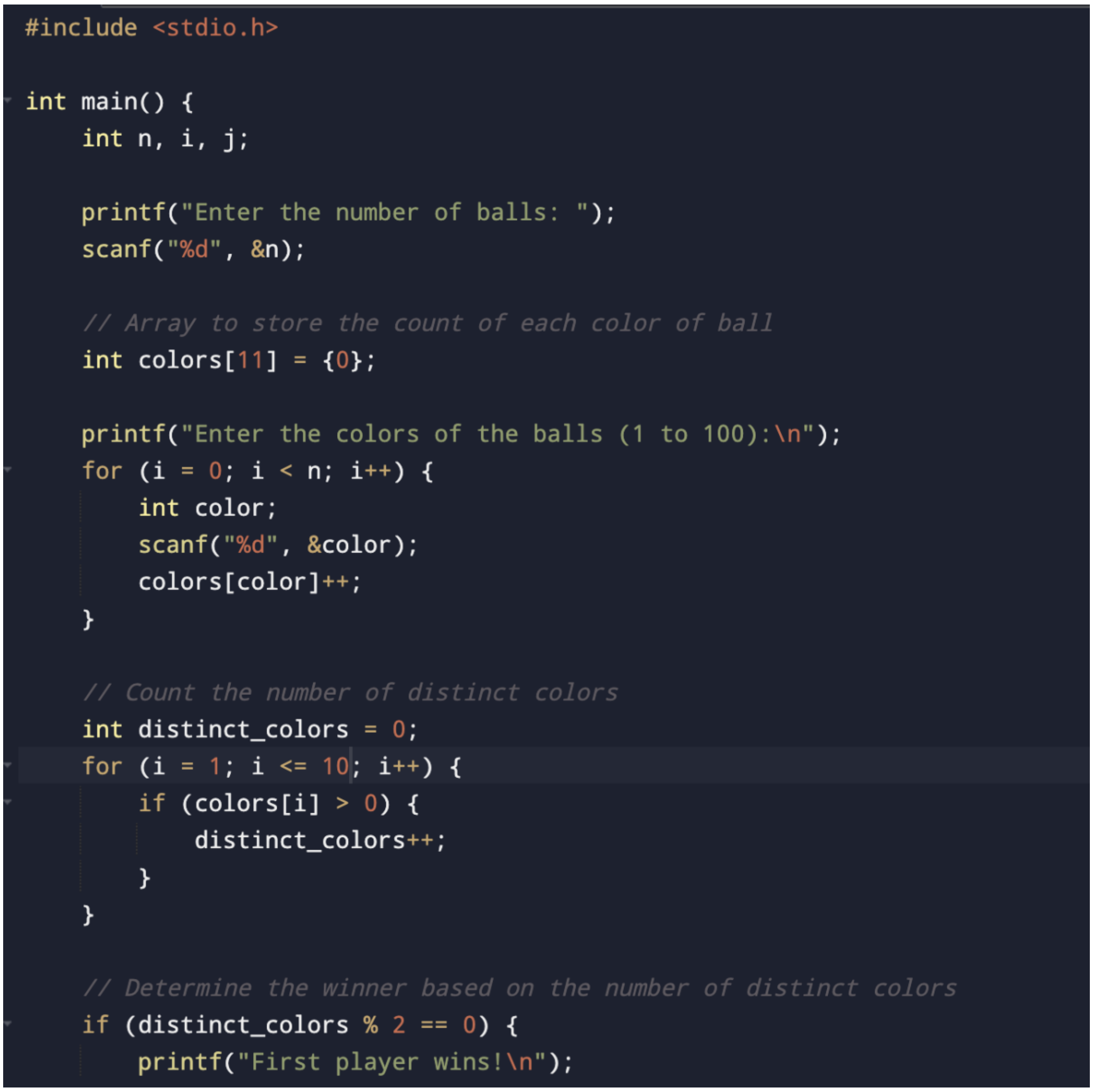


```
1. Add new gadget at the end
2. Add new gadget at the front
3. Search for a gadget
4. Find gadget with maximum price
5. Delete a gadget
6. Exit
Enter your choice: 3
Enter the name of the gadget you want to search for: A
Gadget found:
Name: A
Brand: B
Price: 500.00
```

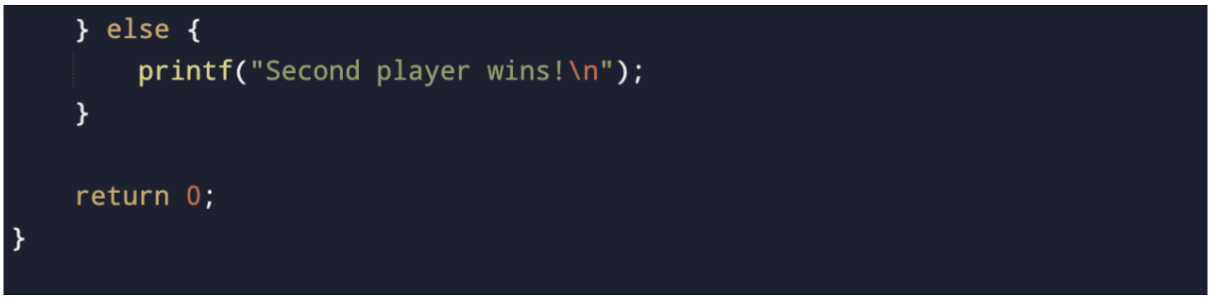


```
1. Add new gadget at the end
2. Add new gadget at the front
3. Search for a gadget
4. Find gadget with maximum price
5. Delete a gadget
6. Exit
Enter your choice: 4
Gadget with maximum price:
Name: H
Brand: I
Price: 600.00
```

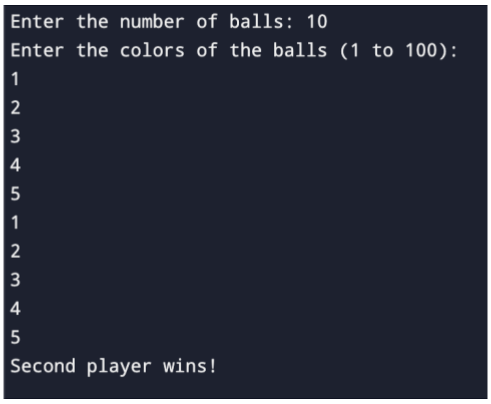
2. There are n balls of different colors in a basket. Two players are playing a game with these balls to make all the balls the same color. One player can play at a time. For each move, a player picks two different color balls and makes them into one color. If there are no two different colored balls in the basket, the player cannot play and he will lose the game. Winner of the game is whoever makes the entire balls the same. Write a C program to simulate this game and find the winner. [Hint: if the count of distinct elements is even, the first player always wins, else the second player wins. Trace this with an example.]



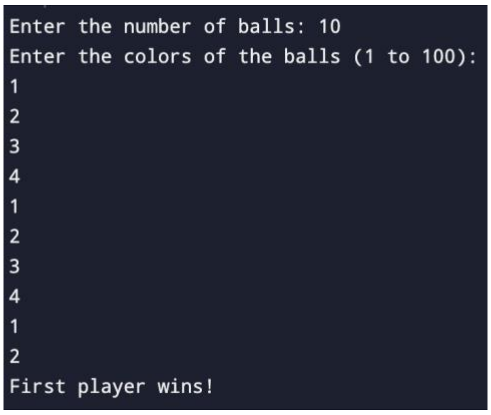
```c
#include <stdio.h>

int main() {
    int n, i, j;

    printf("Enter the number of balls: ");
    scanf("%d", &n);

    // Array to store the count of each color of ball
    int colors[11] = {0};

    printf("Enter the colors of the balls (1 to 100):\n");
    for (i = 0; i < n; i++) {
        int color;
        scanf("%d", &color);
        colors[color]++;
    }

    // Count the number of distinct colors
    int distinct_colors = 0;
    for (i = 1; i <= 10; i++) {
        if (colors[i] > 0) {
            distinct_colors++;
        }
    }

    // Determine the winner based on the number of distinct colors
    if (distinct_colors % 2 == 0) {
        printf("First player wins!\n");
```



```c
    } else {
        printf("Second player wins!\n");
    }

    return 0;
}
```

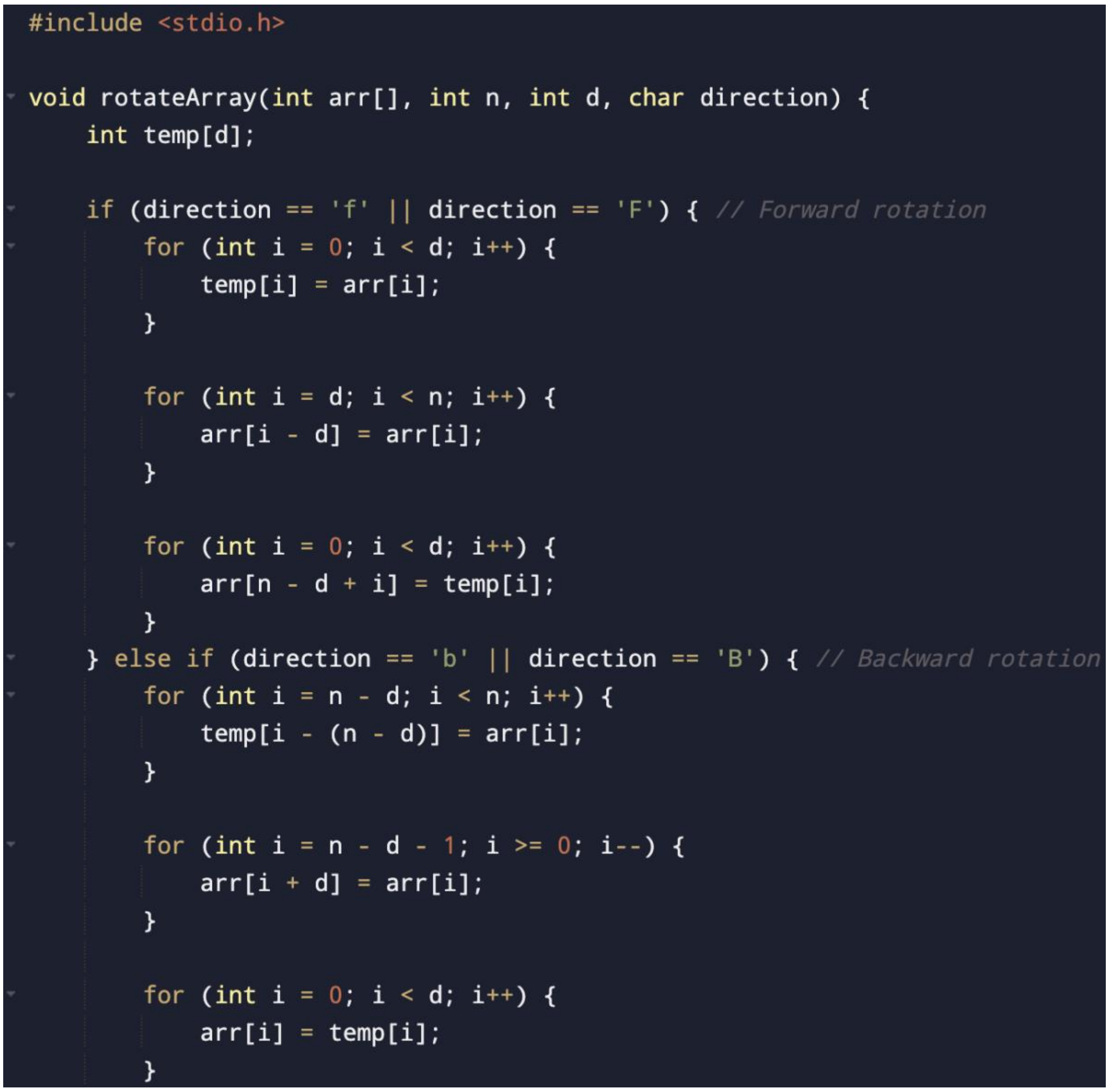


```
Enter the number of balls: 10
Enter the colors of the balls (1 to 100):
1
2
3
4
5
1
2
3
4
5
Second player wins!
```

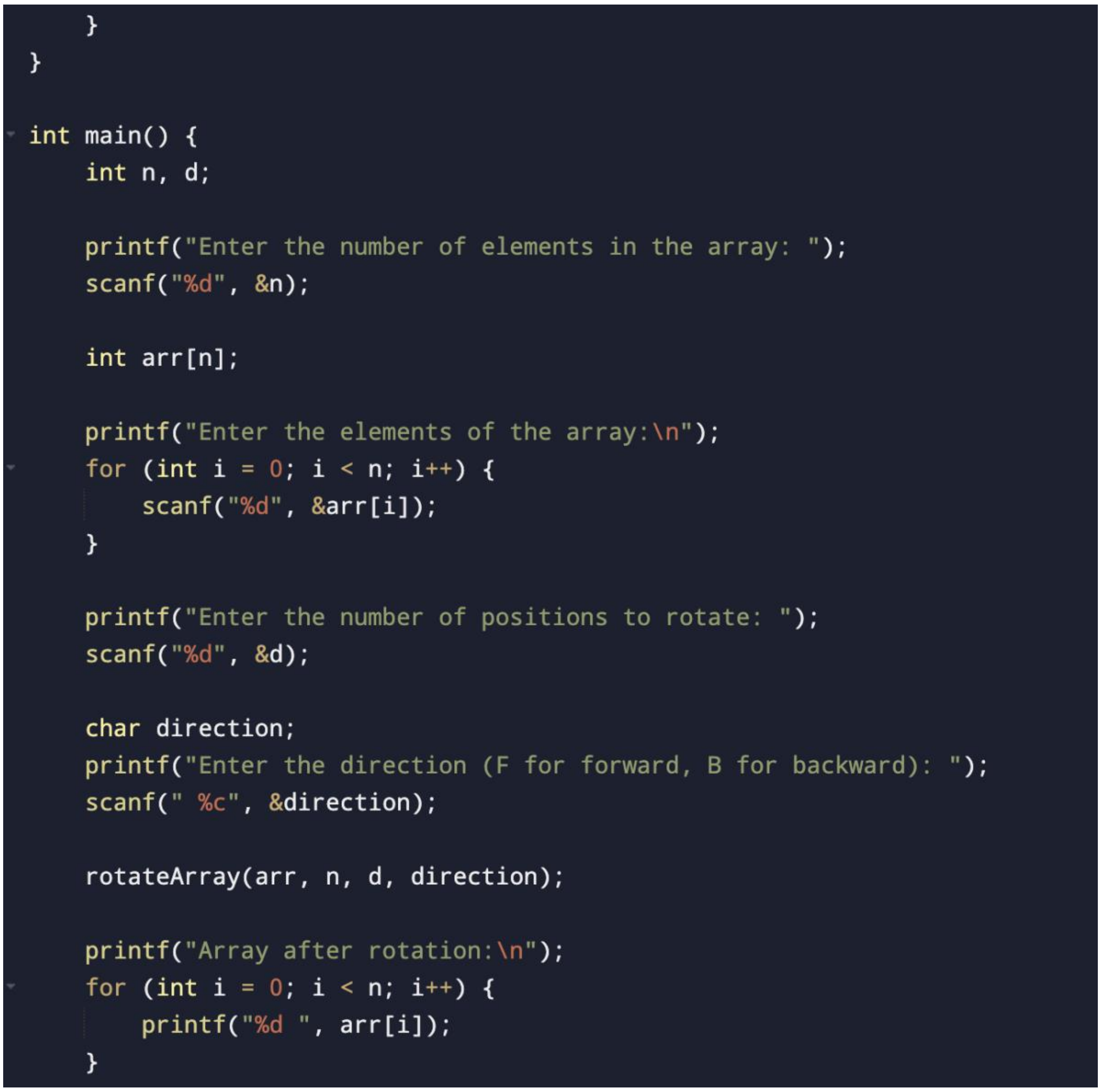


```
Enter the number of balls: 10
Enter the colors of the balls (1 to 100):
1
2
3
4
1
2
3
4
1
2
First player wins!
```

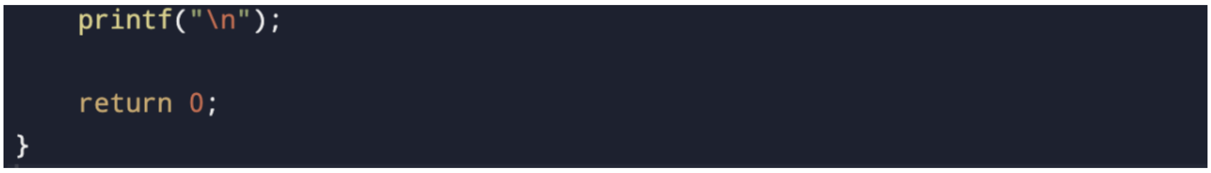
3. Rotate an array by n positions by getting choice from the user for the directions either forward or backward.



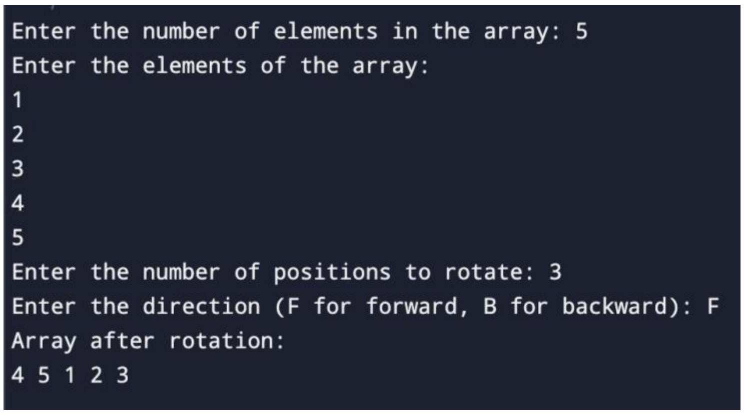
```c
#include <stdio.h>

void rotateArray(int arr[], int n, int d, char direction) {
    int temp[d];

    if (direction == 'f' || direction == 'F') { // Forward rotation
        for (int i = 0; i < d; i++) {
            temp[i] = arr[i];
        }

        for (int i = d; i < n; i++) {
            arr[i - d] = arr[i];
        }

        for (int i = 0; i < d; i++) {
            arr[n - d + i] = temp[i];
        }
    } else if (direction == 'b' || direction == 'B') { // Backward rotation
        for (int i = n - d; i < n; i++) {
            temp[i - (n - d)] = arr[i];
        }

        for (int i = n - d - 1; i >= 0; i--) {
            arr[i + d] = arr[i];
        }

        for (int i = 0; i < d; i++) {
            arr[i] = temp[i];
        }
```



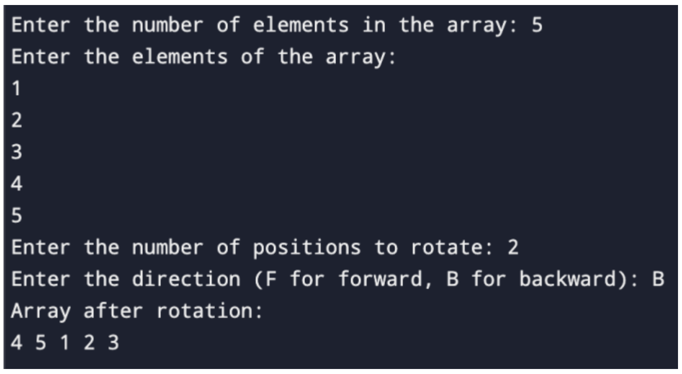
```c
    }
}

int main() {
    int n, d;

    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);

    int arr[n];

    printf("Enter the elements of the array:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter the number of positions to rotate: ");
    scanf("%d", &d);

    char direction;
    printf("Enter the direction (F for forward, B for backward): ");
    scanf(" %c", &direction);

    rotateArray(arr, n, d, direction);

    printf("Array after rotation:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
```



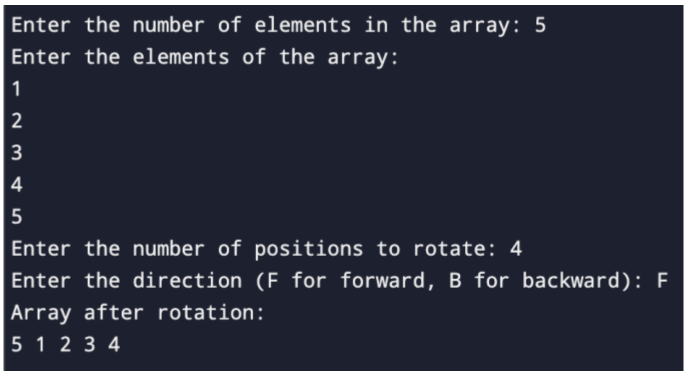
```c
    printf("\n");

    return 0;
}
```



```
Enter the number of elements in the array: 5
Enter the elements of the array:
1
2
3
4
5
Enter the number of positions to rotate: 3
Enter the direction (F for forward, B for backward): F
Array after rotation:
4 5 1 2 3
```

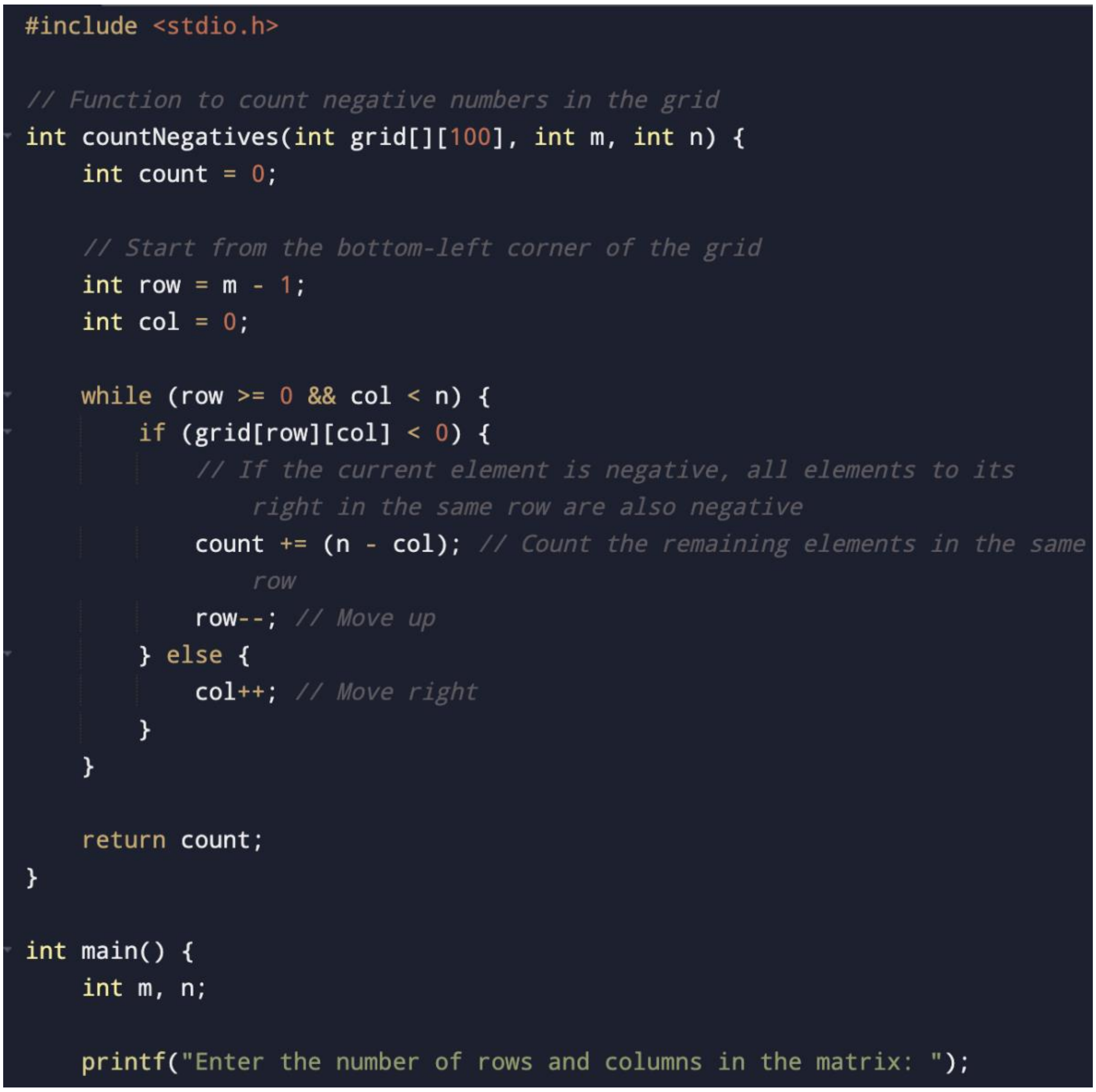


```
Enter the number of elements in the array: 5
Enter the elements of the array:
1
2
3
4
5
Enter the number of positions to rotate: 2
Enter the direction (F for forward, B for backward): B
Array after rotation:
4 5 1 2 3
```

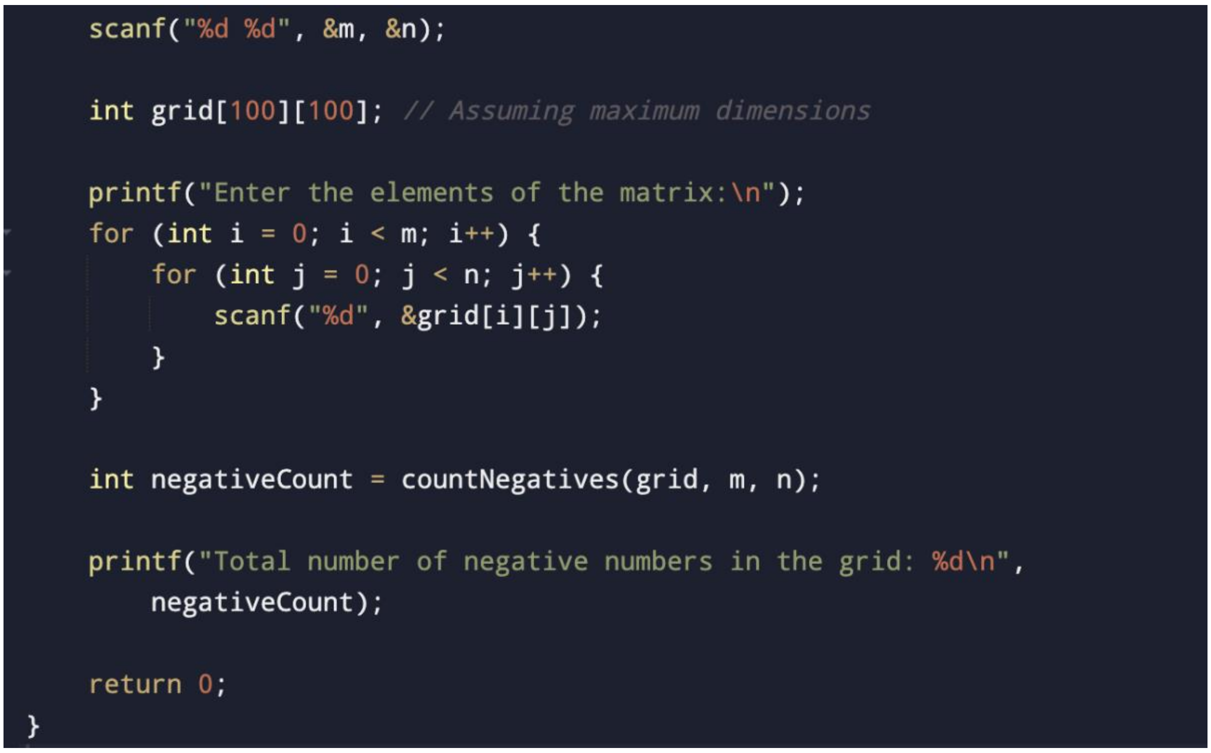


```
Enter the number of elements in the array: 5
Enter the elements of the array:
1
2
3
4
5
Enter the number of positions to rotate: 4
Enter the direction (F for forward, B for backward): F
Array after rotation:
5 1 2 3 4
```

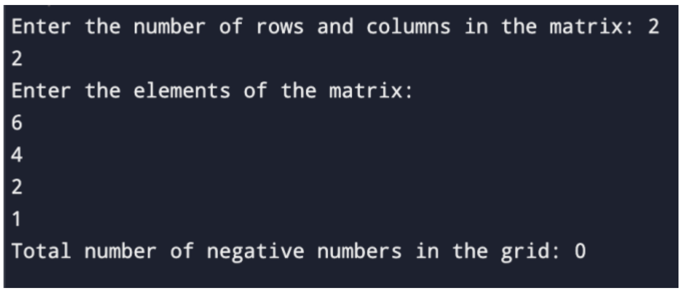
4. Given a m x n matrix grid which is sorted in non-increasing order both row-wise and column-wise, write a C program that uses a function to return the total number of negative numbers in the grid.



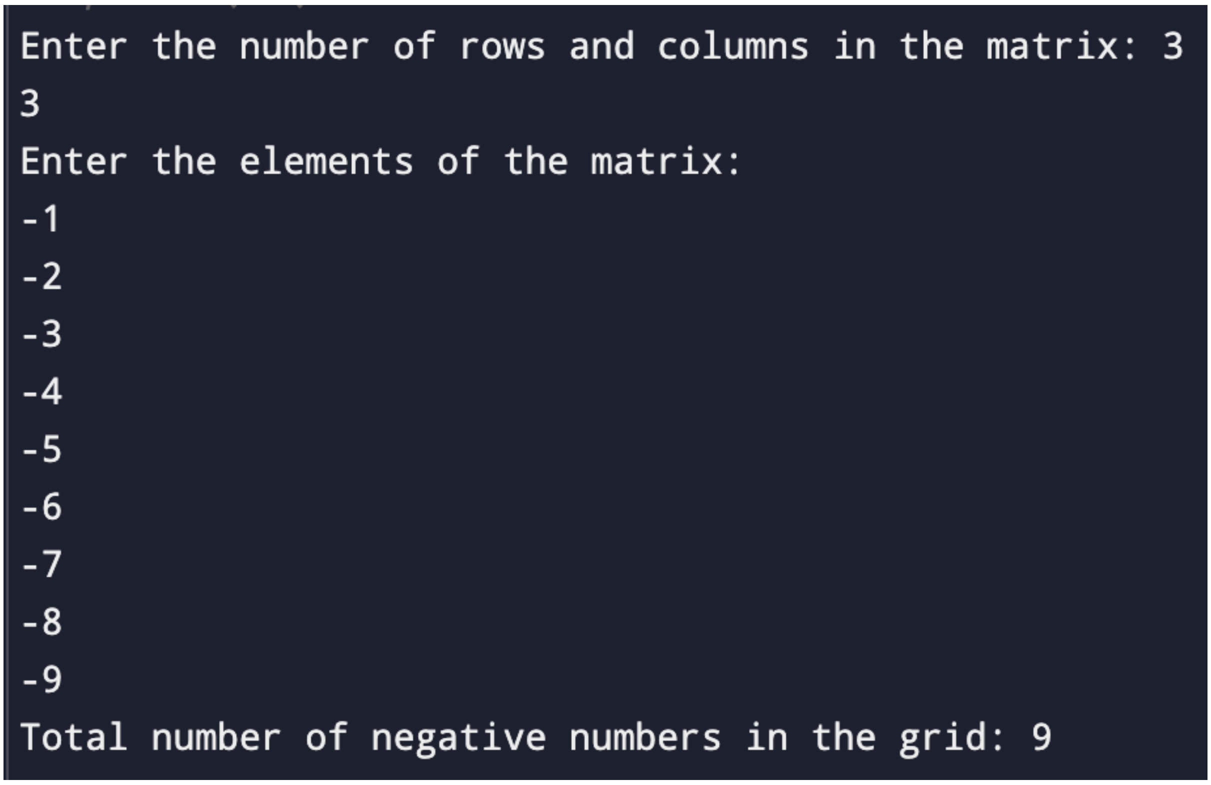
```c
#include <stdio.h>

// Function to count negative numbers in the grid
int countNegatives(int grid[][100], int m, int n) {
    int count = 0;

    // Start from the bottom-left corner of the grid
    int row = m - 1;
    int col = 0;

    while (row >= 0 && col < n) {
        if (grid[row][col] < 0) {
            // If the current element is negative, all elements to its
                right in the same row are also negative
            count += (n - col); // Count the remaining elements in the same
                row
            row--; // Move up
        } else {
            col++; // Move right
        }
    }

    return count;
}

int main() {
    int m, n;

    printf("Enter the number of rows and columns in the matrix: ");
```



```c
    scanf("%d %d", &m, &n);

    int grid[100][100]; // Assuming maximum dimensions

    printf("Enter the elements of the matrix:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &grid[i][j]);
        }
    }

    int negativeCount = countNegatives(grid, m, n);

    printf("Total number of negative numbers in the grid: %d\n",
        negativeCount);

    return 0;
}
```

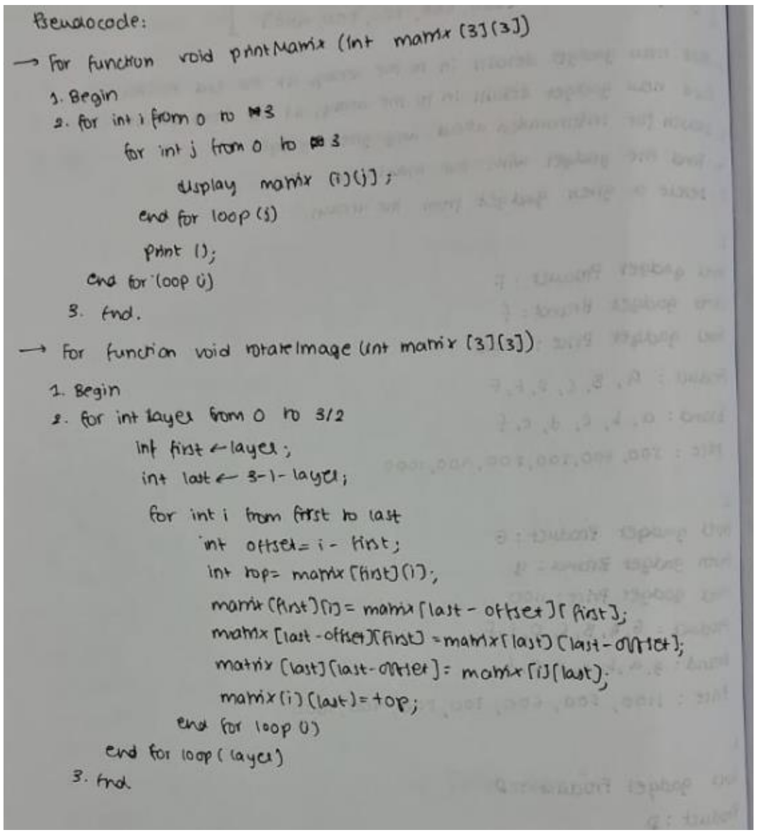


```
Enter the number of rows and columns in the matrix: 2
2
Enter the elements of the matrix:
6
4
2
1
Total number of negative numbers in the grid: 0
```

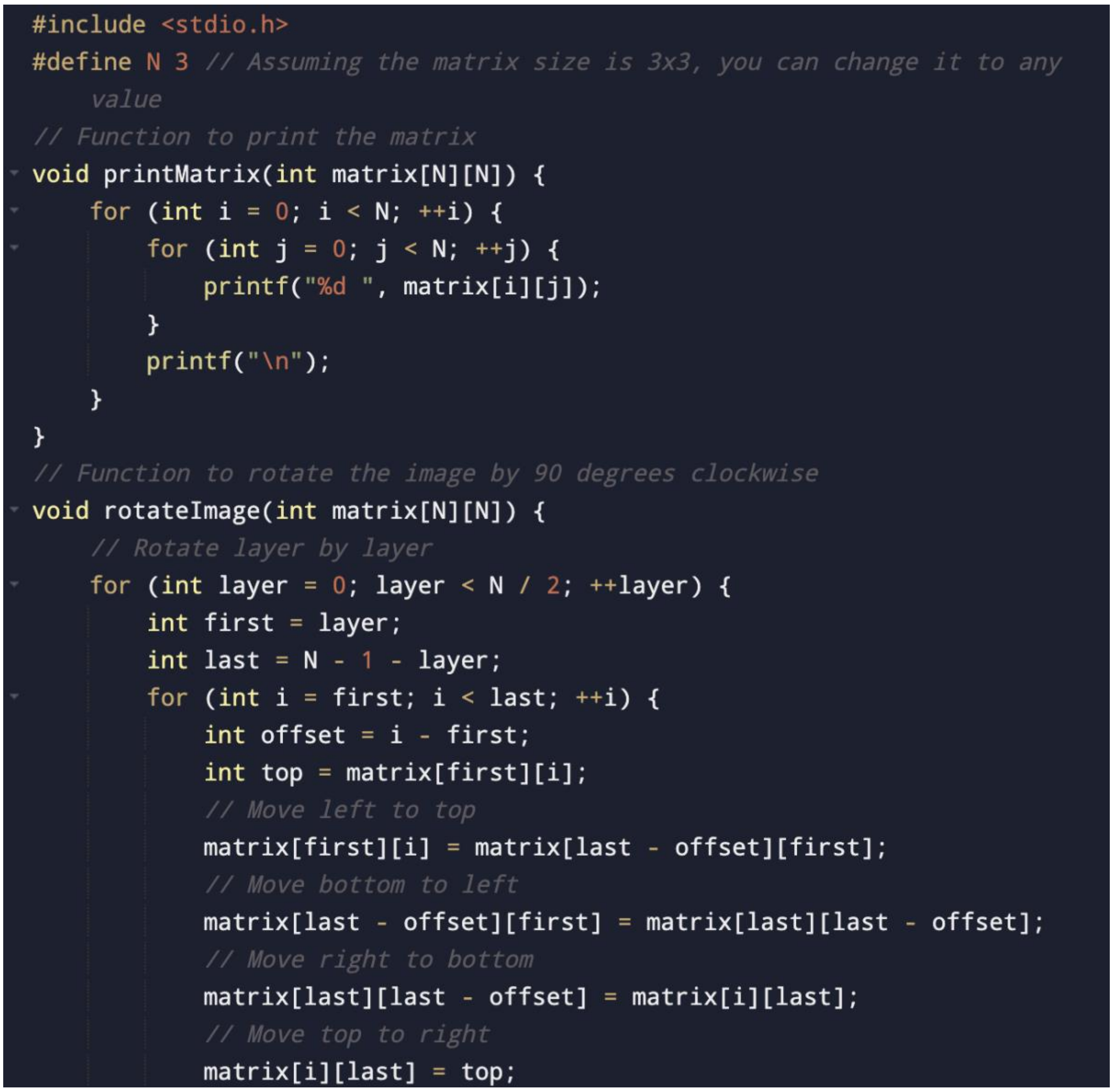


```
Enter the number of rows and columns in the matrix: 3
3
Enter the elements of the matrix:
-1
-2
-3
-4
-5
-6
-7
-8
-9
Total number of negative numbers in the grid: 9
```

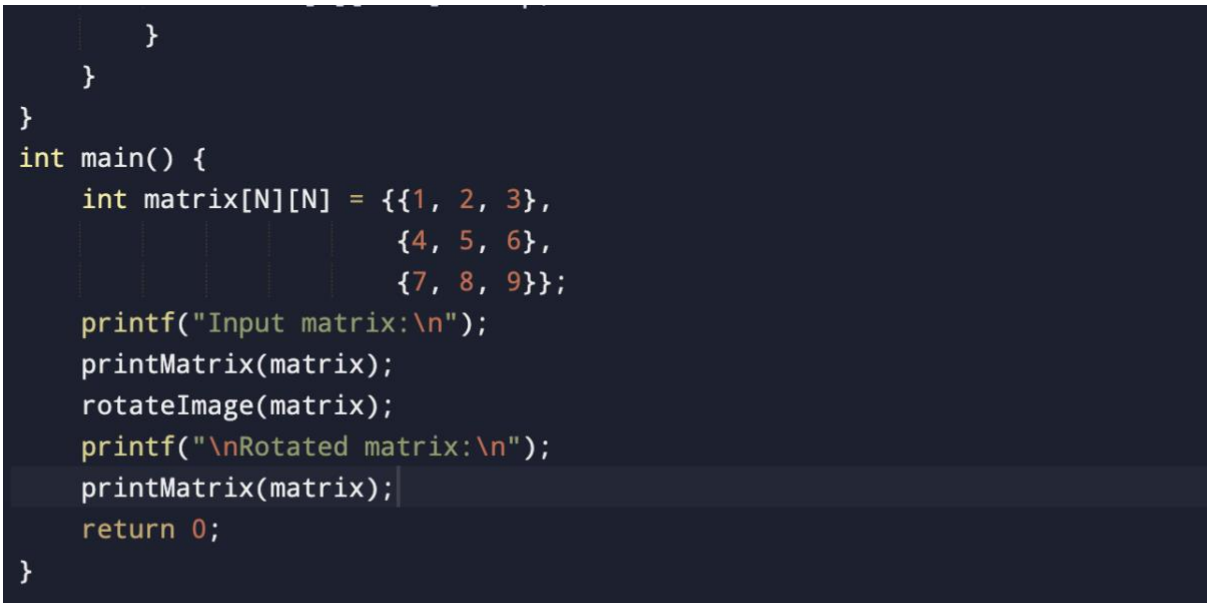
5. Given an n x n 2D matrix representing an image, rotate the image by 90 degrees(clockwise). You have to rotate the image in-place, which means you should modify the input 2D matrix directly. Do not allocate another 2D matrix and do the rotation.



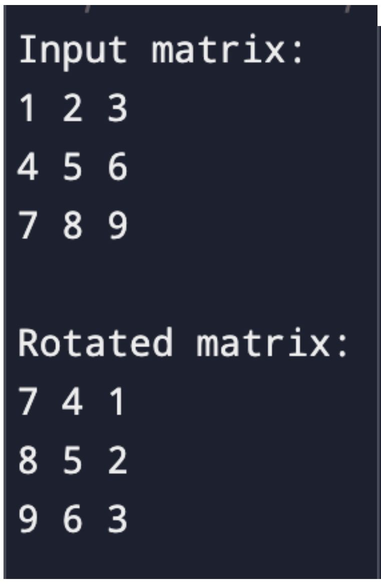
```
Pseudocode:
→ For function void printMatrix (int matrix [3][3])
  1. Begin
  2. for int i from 0 to 3
          for int j from 0 to 3
              display matrix [i][j] ;
          end for loop (j)
          print ();
      end for loop (i)
    3. End.
→ For function void rotateImage (int matrix [3][3])
  1. Begin
  2. for int layer from 0 to 3/2
          int first ← layer ;
          int last ← 3-1- layer;
          for int i from first to last
               int offset = i - first;
               int top= matrix [first][i];
               matrix [first][i] = matrix [last - offset][first];
               matrix [last - offset][first] = matrix [last][last - offset];
               matrix [last][last - offset] = matrix [i][last];
               matrix [i][last] = top;
          end for loop (i)
      end for loop (layer)
    3. End
```



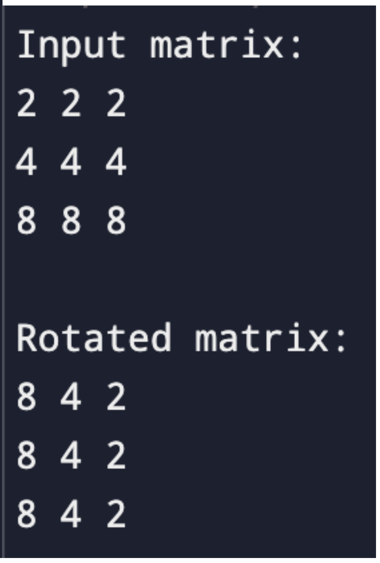
```c
#include <stdio.h>
#define N 3 // Assuming the matrix size is 3x3, you can change it to any value
// Function to print the matrix
void printMatrix(int matrix[N][N]) {
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}
// Function to rotate the image by 90 degrees clockwise
void rotateImage(int matrix[N][N]) {
    // Rotate layer by layer
    for (int layer = 0; layer < N / 2; ++layer) {
        int first = layer;
        int last = N - 1 - layer;
        for (int i = first; i < last; ++i) {
            int offset = i - first;
            int top = matrix[first][i];
            // Move left to top
            matrix[first][i] = matrix[last - offset][first];
            // Move bottom to left
            matrix[last - offset][first] = matrix[last][last - offset];
            // Move right to bottom
            matrix[last][last - offset] = matrix[i][last];
            // Move top to right
            matrix[i][last] = top;
```



```c
        }
    }
}
int main() {
    int matrix[N][N] = {{1, 2, 3},
                        {4, 5, 6},
                        {7, 8, 9}};
    printf("Input matrix:\n");
    printMatrix(matrix);
    rotateImage(matrix);
    printf("\nRotated matrix:\n");
    printMatrix(matrix);
    return 0;
}
```



```
Input matrix:
1 2 3
4 5 6
7 8 9

Rotated matrix:
7 4 1
8 5 2
9 6 3
```



```
Input matrix:
2 2 2
4 4 4
8 8 8

Rotated matrix:
8 4 2
8 4 2
8 4 2
```