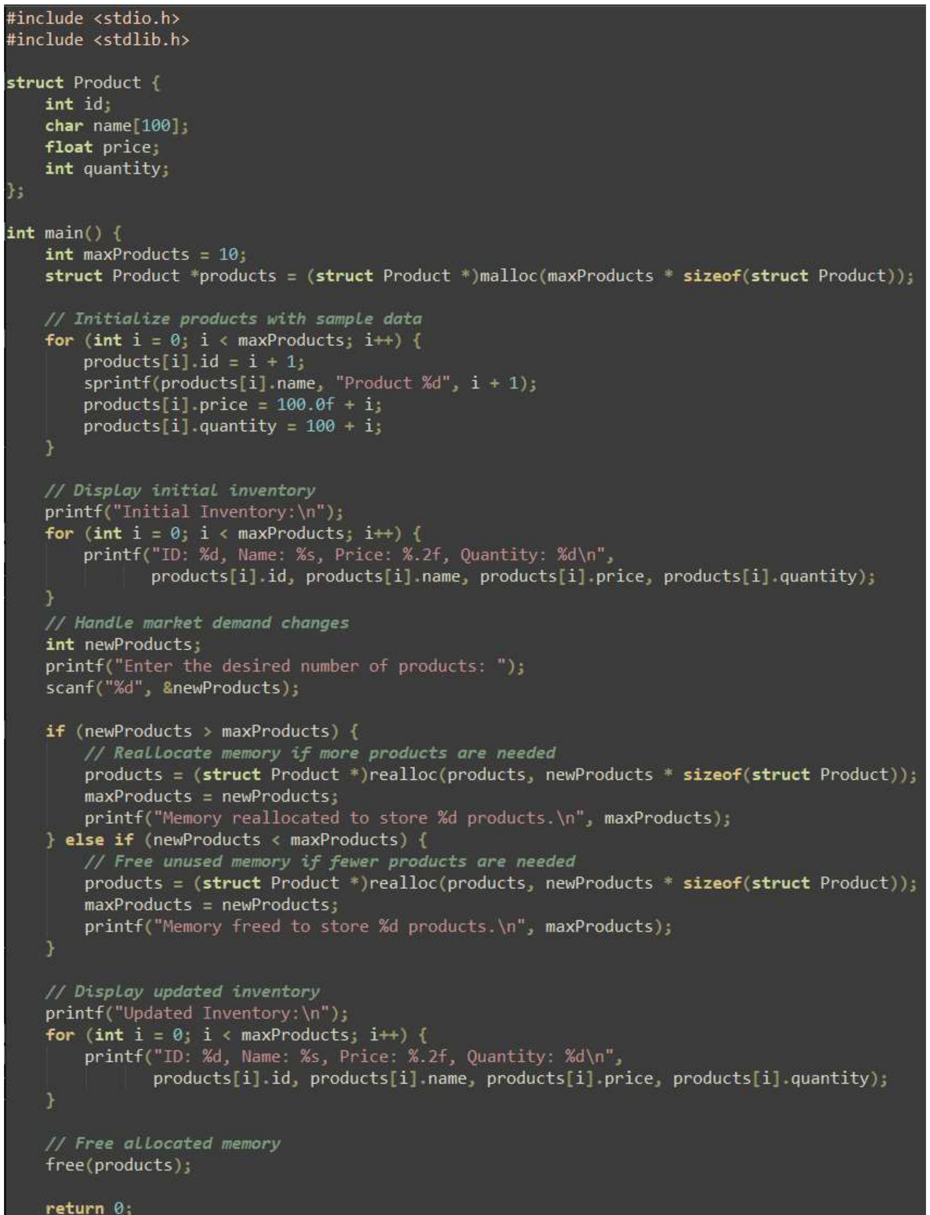
DSA ASSIGNMENT

By Shreya Ravi K (RA2311003050020)

Question - 1

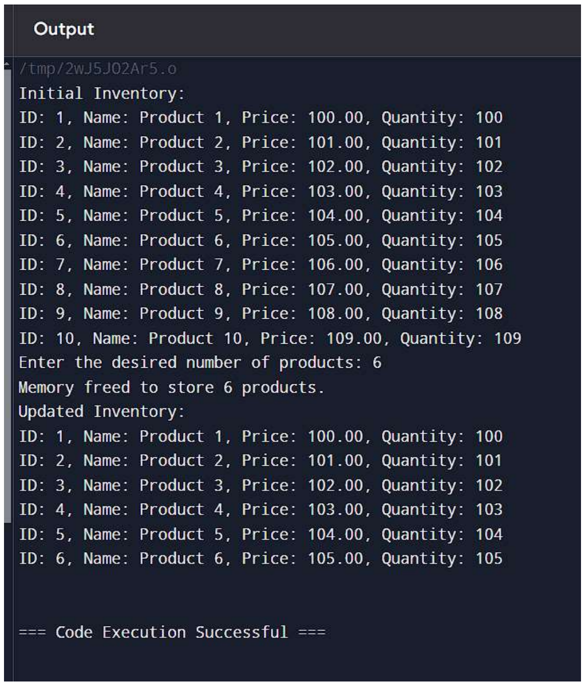
Mr. Ram is developing a software application to manage his inventory, which initially can hold a maximum of ten product packets. He allocates memory for ten packets at the start. However, as the market demand changes, he needs to dynamically update the allocated memory to store more or fewer packets without wasting memory or running out of space. Explain the concept of dynamic memory allocation and how Mr. Ram can effectively manage his inventory size using this concept.

Answer:

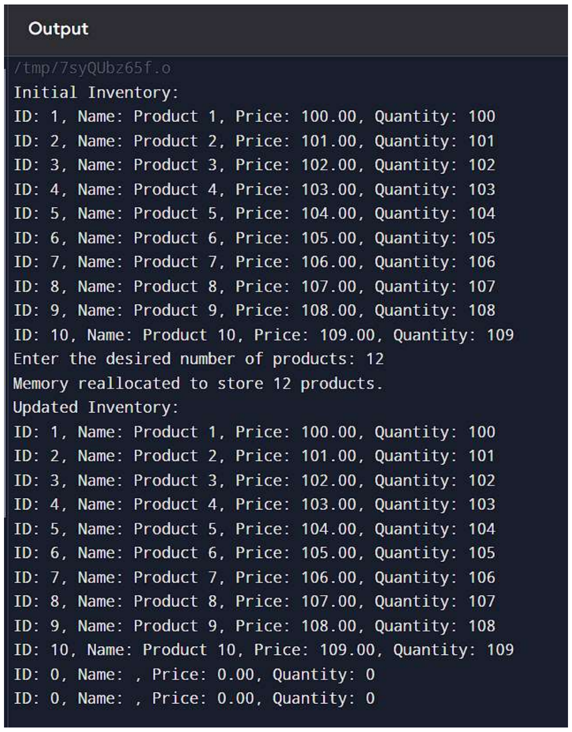


```c
#include <stdio.h>
#include <stdlib.h>

struct Product {
    int id;
    char name[100];
    float price;
    int quantity;
};

int main() {
    int maxProducts = 10;
    struct Product *products = (struct Product *)malloc(maxProducts * sizeof(struct Product));

    // Initialize products with sample data
    for (int i = 0; i < maxProducts; i++) {
        products[i].id = i + 1;
        sprintf(products[i].name, "Product %d", i + 1);
        products[i].price = 100.0f + i;
        products[i].quantity = 100 + i;
    }

    // Display initial inventory
    printf("Initial Inventory:\n");
    for (int i = 0; i < maxProducts; i++) {
        printf("ID: %d, Name: %s, Price: %.2f, Quantity: %d\n",
               products[i].id, products[i].name, products[i].price, products[i].quantity);
    }
    // Handle market demand changes
    int newProducts;
    printf("Enter the desired number of products: ");
    scanf("%d", &newProducts);

    if (newProducts > maxProducts) {
        // Reallocate memory if more products are needed
        products = (struct Product *)realloc(products, newProducts * sizeof(struct Product));
        maxProducts = newProducts;
        printf("Memory reallocated to store %d products.\n", maxProducts);
    } else if (newProducts < maxProducts) {
        // Free unused memory if fewer products are needed
        products = (struct Product *)realloc(products, newProducts * sizeof(struct Product));
        maxProducts = newProducts;
        printf("Memory freed to store %d products.\n", maxProducts);
    }

    // Display updated inventory
    printf("Updated Inventory:\n");
    for (int i = 0; i < maxProducts; i++) {
        printf("ID: %d, Name: %s, Price: %.2f, Quantity: %d\n",
               products[i].id, products[i].name, products[i].price, products[i].quantity);
    }

    // Free allocated memory
    free(products);

    return 0;
```

Output:



```
Output
/tmp/2wJ5JO2Ar5.o
Initial Inventory:
ID: 1, Name: Product 1, Price: 100.00, Quantity: 100
ID: 2, Name: Product 2, Price: 101.00, Quantity: 101
ID: 3, Name: Product 3, Price: 102.00, Quantity: 102
ID: 4, Name: Product 4, Price: 103.00, Quantity: 103
ID: 5, Name: Product 5, Price: 104.00, Quantity: 104
ID: 6, Name: Product 6, Price: 105.00, Quantity: 105
ID: 7, Name: Product 7, Price: 106.00, Quantity: 106
ID: 8, Name: Product 8, Price: 107.00, Quantity: 107
ID: 9, Name: Product 9, Price: 108.00, Quantity: 108
ID: 10, Name: Product 10, Price: 109.00, Quantity: 109
Enter the desired number of products: 6
Memory freed to store 6 products.
Updated Inventory:
ID: 1, Name: Product 1, Price: 100.00, Quantity: 100
ID: 2, Name: Product 2, Price: 101.00, Quantity: 101
ID: 3, Name: Product 3, Price: 102.00, Quantity: 102
ID: 4, Name: Product 4, Price: 103.00, Quantity: 103
ID: 5, Name: Product 5, Price: 104.00, Quantity: 104
ID: 6, Name: Product 6, Price: 105.00, Quantity: 105


=== Code Execution Successful ===
```

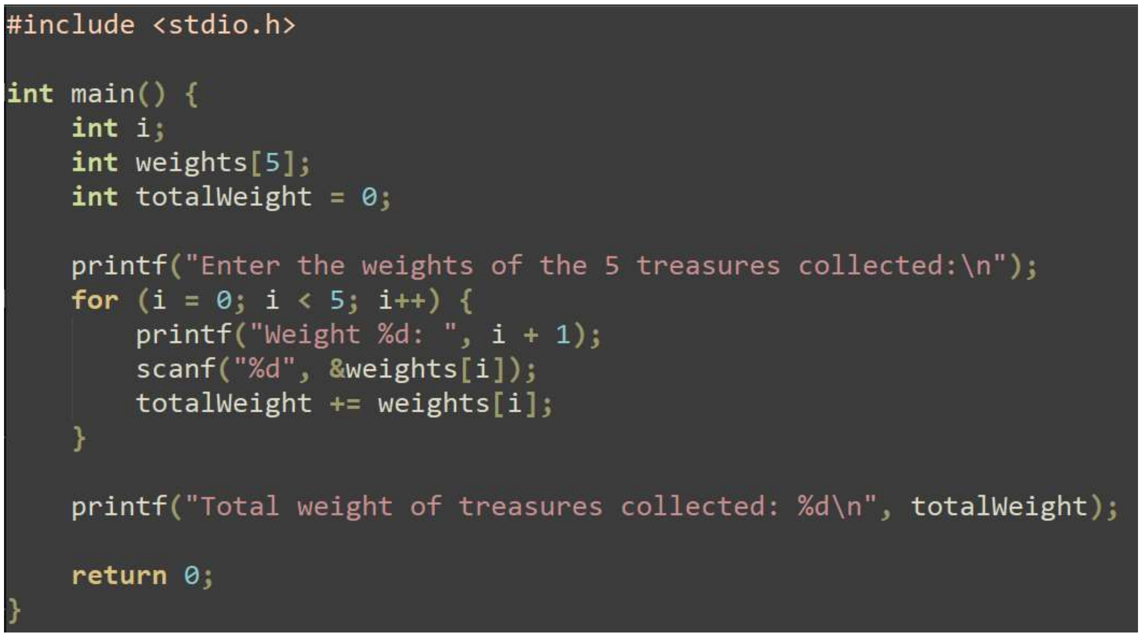


```
Output
/tmp/7syQUbz65f.o
Initial Inventory:
ID: 1, Name: Product 1, Price: 100.00, Quantity: 100
ID: 2, Name: Product 2, Price: 101.00, Quantity: 101
ID: 3, Name: Product 3, Price: 102.00, Quantity: 102
ID: 4, Name: Product 4, Price: 103.00, Quantity: 103
ID: 5, Name: Product 5, Price: 104.00, Quantity: 104
ID: 6, Name: Product 6, Price: 105.00, Quantity: 105
ID: 7, Name: Product 7, Price: 106.00, Quantity: 106
ID: 8, Name: Product 8, Price: 107.00, Quantity: 107
ID: 9, Name: Product 9, Price: 108.00, Quantity: 108
ID: 10, Name: Product 10, Price: 109.00, Quantity: 109
Enter the desired number of products: 12
Memory reallocated to store 12 products.
Updated Inventory:
ID: 1, Name: Product 1, Price: 100.00, Quantity: 100
ID: 2, Name: Product 2, Price: 101.00, Quantity: 101
ID: 3, Name: Product 3, Price: 102.00, Quantity: 102
ID: 4, Name: Product 4, Price: 103.00, Quantity: 103
ID: 5, Name: Product 5, Price: 104.00, Quantity: 104
ID: 6, Name: Product 6, Price: 105.00, Quantity: 105
ID: 7, Name: Product 7, Price: 106.00, Quantity: 106
ID: 8, Name: Product 8, Price: 107.00, Quantity: 107
ID: 9, Name: Product 9, Price: 108.00, Quantity: 108
ID: 10, Name: Product 10, Price: 109.00, Quantity: 109
ID: 0, Name: , Price: 0.00, Quantity: 0
ID: 0, Name: , Price: 0.00, Quantity: 0
```

Question 2:

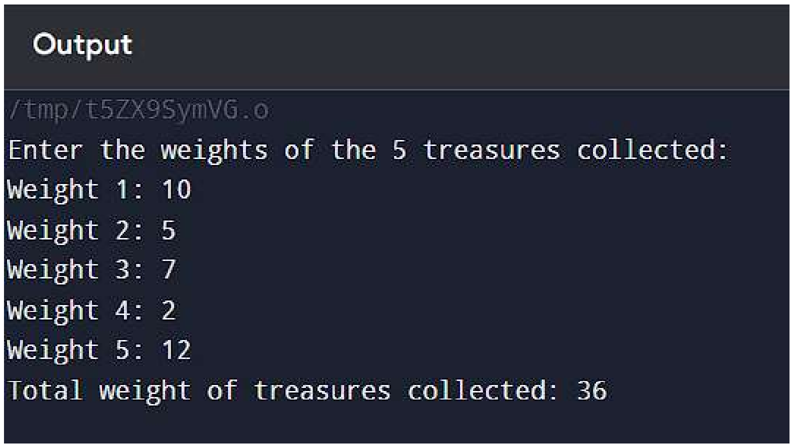
Mr. John is playing the game Subway Surfers. The game has a total of five treasures with different weights, that he needs to collect. Write a C program to count the total number of weights he collected from the treasures during the game.

Answer:



```c
#include <stdio.h>

int main() {
    int i;
    int weights[5];
    int totalWeight = 0;

    printf("Enter the weights of the 5 treasures collected:\n");
    for (i = 0; i < 5; i++) {
        printf("Weight %d: ", i + 1);
        scanf("%d", &weights[i]);
        totalWeight += weights[i];
    }

    printf("Total weight of treasures collected: %d\n", totalWeight);

    return 0;
}
```

Output:

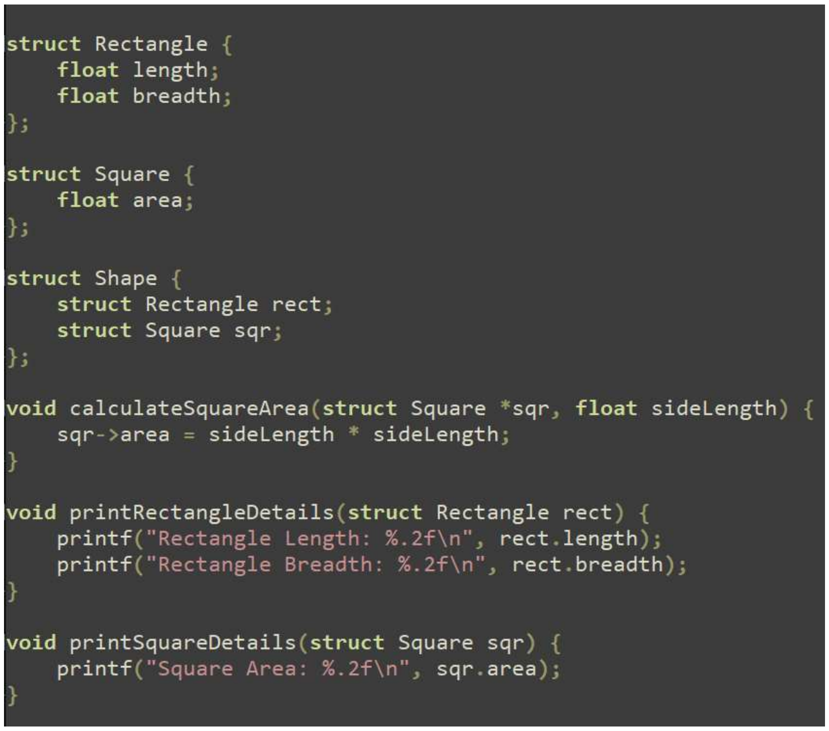


```
Output
/tmp/t5ZX9SymVG.o
Enter the weights of the 5 treasures collected:
Weight 1: 10
Weight 2: 5
Weight 3: 7
Weight 4: 2
Weight 5: 12
Total weight of treasures collected: 36
```

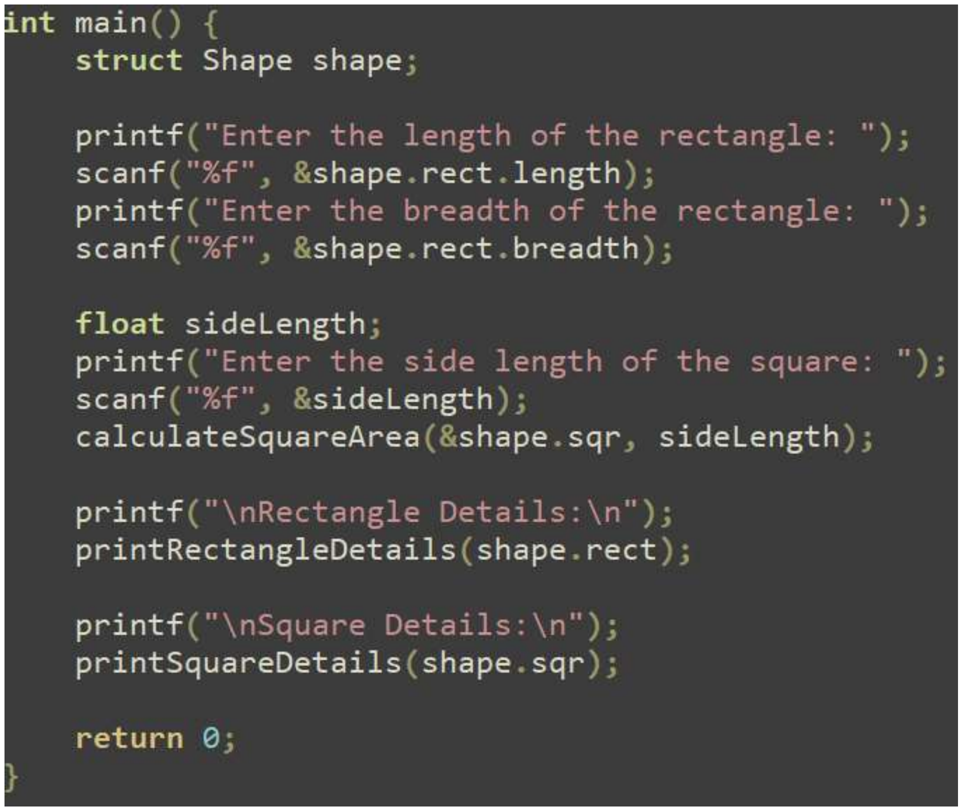
Question 3:

Consider an object Shape that encompasses both a Square and a Rectangle as the data members. The Square object should have an attribute for its area, while the Rectangle object should have attributes for length and breadth. Identify the most suitable data structures for this scenario and write a C program to define the structure and demonstrate their usage.

Answer:

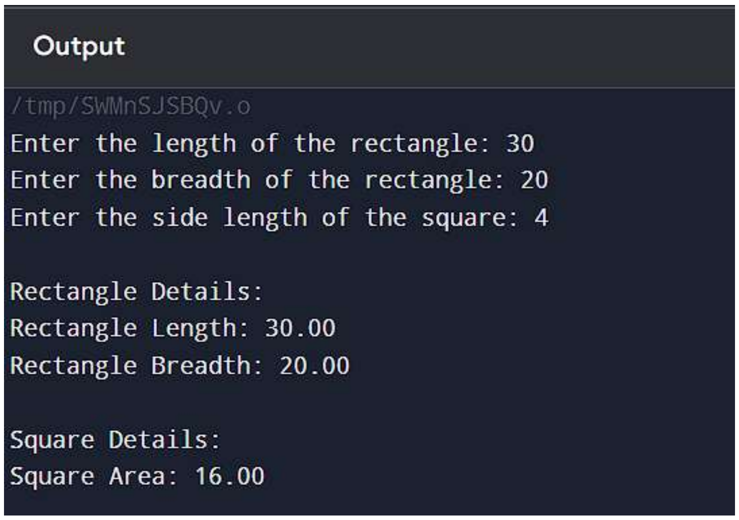


```c
struct Rectangle {
    float length;
    float breadth;
};

struct Square {
    float area;
};

struct Shape {
    struct Rectangle rect;
    struct Square sqr;
};

void calculateSquareArea(struct Square *sqr, float sideLength) {
    sqr->area = sideLength * sideLength;
}

void printRectangleDetails(struct Rectangle rect) {
    printf("Rectangle Length: %.2f\n", rect.length);
    printf("Rectangle Breadth: %.2f\n", rect.breadth);
}

void printSquareDetails(struct Square sqr) {
    printf("Square Area: %.2f\n", sqr.area);
}
```



```c
int main() {
    struct Shape shape;

    printf("Enter the length of the rectangle: ");
    scanf("%f", &shape.rect.length);
    printf("Enter the breadth of the rectangle: ");
    scanf("%f", &shape.rect.breadth);

    float sideLength;
    printf("Enter the side length of the square: ");
    scanf("%f", &sideLength);
    calculateSquareArea(&shape.sqr, sideLength);

    printf("\nRectangle Details:\n");
    printRectangleDetails(shape.rect);

    printf("\nSquare Details:\n");
    printSquareDetails(shape.sqr);

    return 0;
}
```

Output:



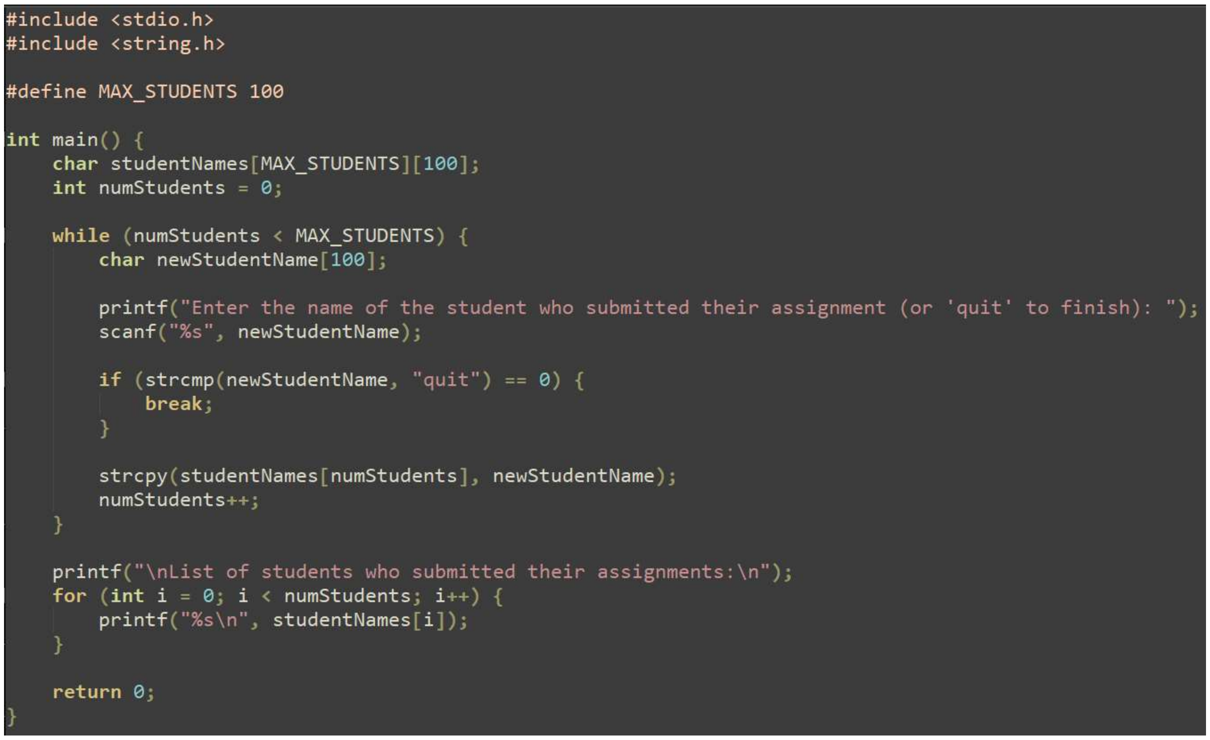
Output

```
/tmp/SWMnSJSBQv.o
Enter the length of the rectangle: 30
Enter the breadth of the rectangle: 20
Enter the side length of the square: 4

Rectangle Details:
Rectangle Length: 30.00
Rectangle Breadth: 20.00

Square Details:
Square Area: 16.00
```

Question 4:

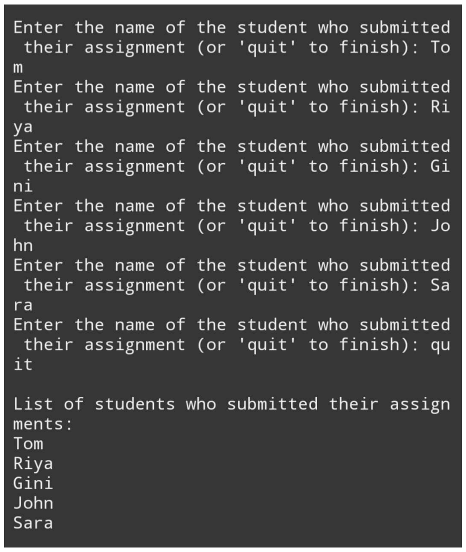
In a classroom, the teacher wants to create a list of students who have submitted their assignments. As students submit their work, the teacher needs to add each student's name to the list in the order of submission. Help the teacher by guiding them on how to use a proper data structure to insert each student's name into the list as they submit their assignment. Write a C program that uses an array to manage the list and demonstrates how to insert new student names into the array.

Answer:



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

#define MAX_STUDENTS 100

int main() {
    char studentNames[MAX_STUDENTS][100];
    int numStudents = 0;

    while (numStudents < MAX_STUDENTS) {
        char newStudentName[100];

        printf("Enter the name of the student who submitted their assignment (or 'quit' to finish): ");
        scanf("%s", newStudentName);

        if (strcmp(newStudentName, "quit") == 0) {
            break;
        }

        strcpy(studentNames[numStudents], newStudentName);
        numStudents++;
    }

    printf("\nList of students who submitted their assignments:\n");
    for (int i = 0; i < numStudents; i++) {
        printf("%s\n", studentNames[i]);
    }

    return 0;
}
```

Output:

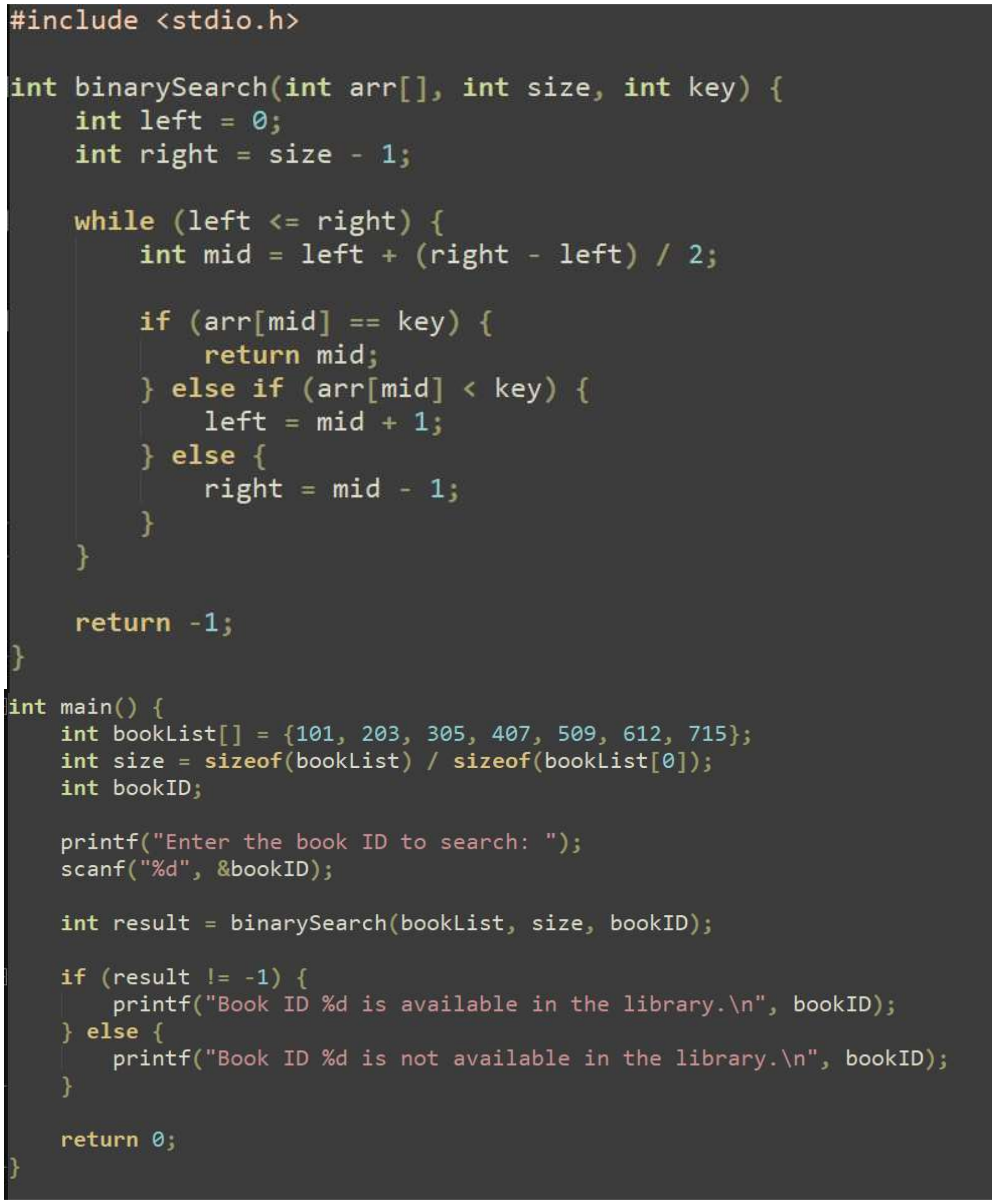


```
Enter the name of the student who submitted
 their assignment (or 'quit' to finish): To
m
Enter the name of the student who submitted
 their assignment (or 'quit' to finish): Ri
ya
Enter the name of the student who submitted
 their assignment (or 'quit' to finish): Gi
ni
Enter the name of the student who submitted
 their assignment (or 'quit' to finish): Jo
hn
Enter the name of the student who submitted
 their assignment (or 'quit' to finish): Sa
ra
Enter the name of the student who submitted
 their assignment (or 'quit' to finish): qu
it

List of students who submitted their assign
ments:
Tom
Riya
Gini
John
Sara
```

Question 5:

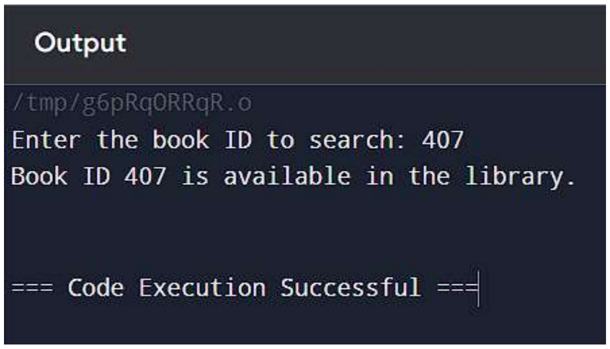
Students need to check the availability of a book in the library based on its ID. Create an ordered list which will contain only book id. Implement a C program to search whether a particular book is available in the list or not.

Answer:



```c
#include <stdio.h>

int binarySearch(int arr[], int size, int key) {
    int left = 0;
    int right = size - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == key) {
            return mid;
        } else if (arr[mid] < key) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    return -1;
}
int main() {
    int bookList[] = {101, 203, 305, 407, 509, 612, 715};
    int size = sizeof(bookList) / sizeof(bookList[0]);
    int bookID;

    printf("Enter the book ID to search: ");
    scanf("%d", &bookID);

    int result = binarySearch(bookList, size, bookID);

    if (result != -1) {
        printf("Book ID %d is available in the library.\n", bookID);
    } else {
        printf("Book ID %d is not available in the library.\n", bookID);
    }

    return 0;
}
```

Output:



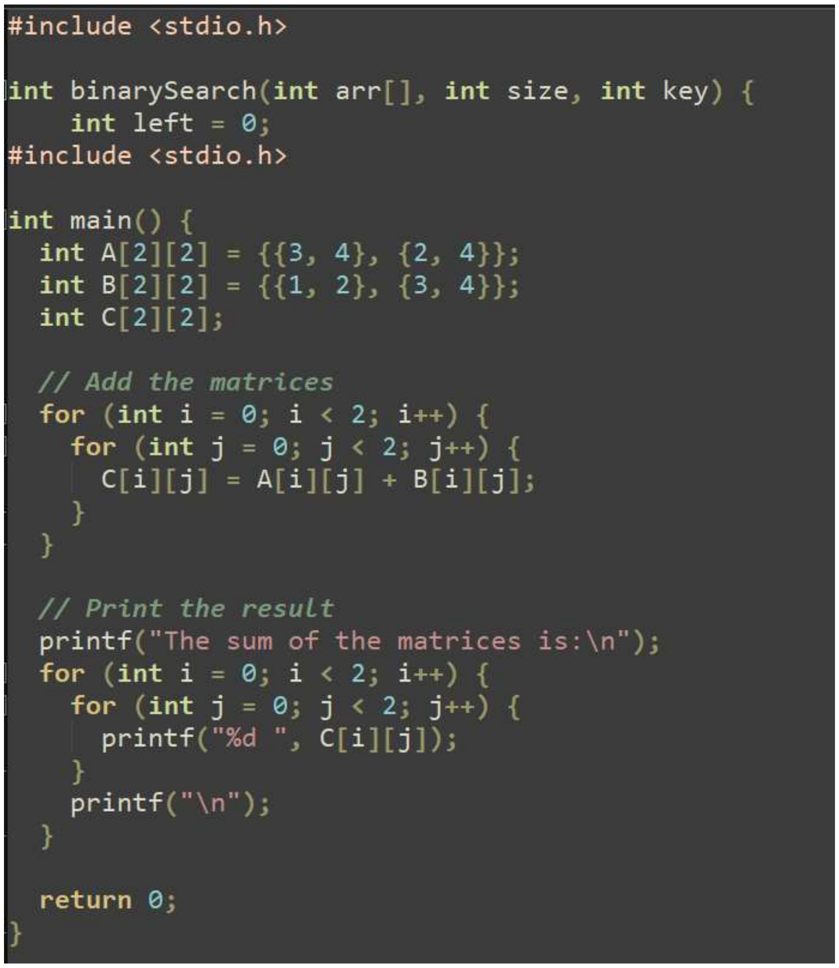
Output

```
/tmp/g6pRqORRqR.o
Enter the book ID to search: 407
Book ID 407 is available in the library.


=== Code Execution Successful ===
```

Question 6:

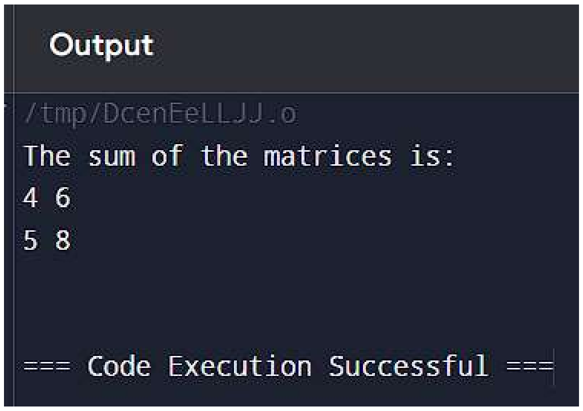
A=[3 4 2 4] B=[1 2 3 4] Write a C program to perform Matrix addition for the above two matrices.

Answer:



```c
#include <stdio.h>

int binarySearch(int arr[], int size, int key) {
    int left = 0;
#include <stdio.h>

int main() {
  int A[2][2] = {{3, 4}, {2, 4}};
  int B[2][2] = {{1, 2}, {3, 4}};
  int C[2][2];

  // Add the matrices
  for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 2; j++) {
      C[i][j] = A[i][j] + B[i][j];
    }
  }

  // Print the result
  printf("The sum of the matrices is:\n");
  for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 2; j++) {
      printf("%d ", C[i][j]);
    }
    printf("\n");
  }

  return 0;
}
```

Output:

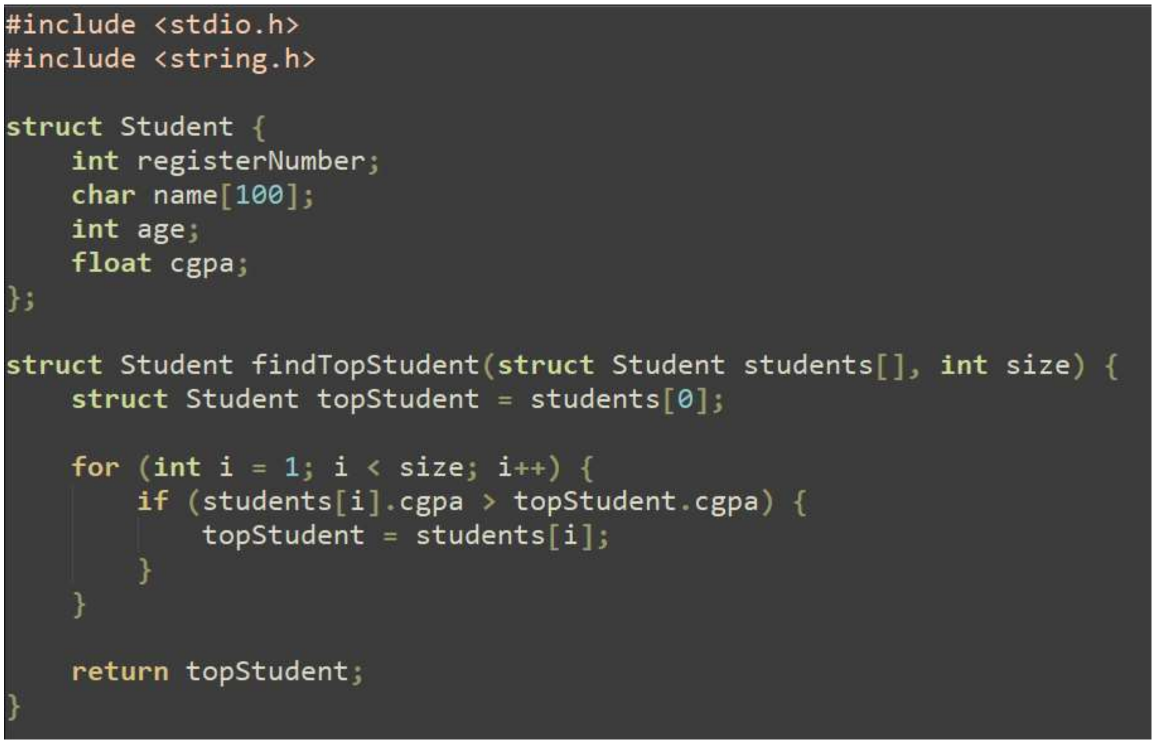


```
Output
/tmp/DcenEeLLJJ.o
The sum of the matrices is:
4 6
5 8


=== Code Execution Successful ===
```

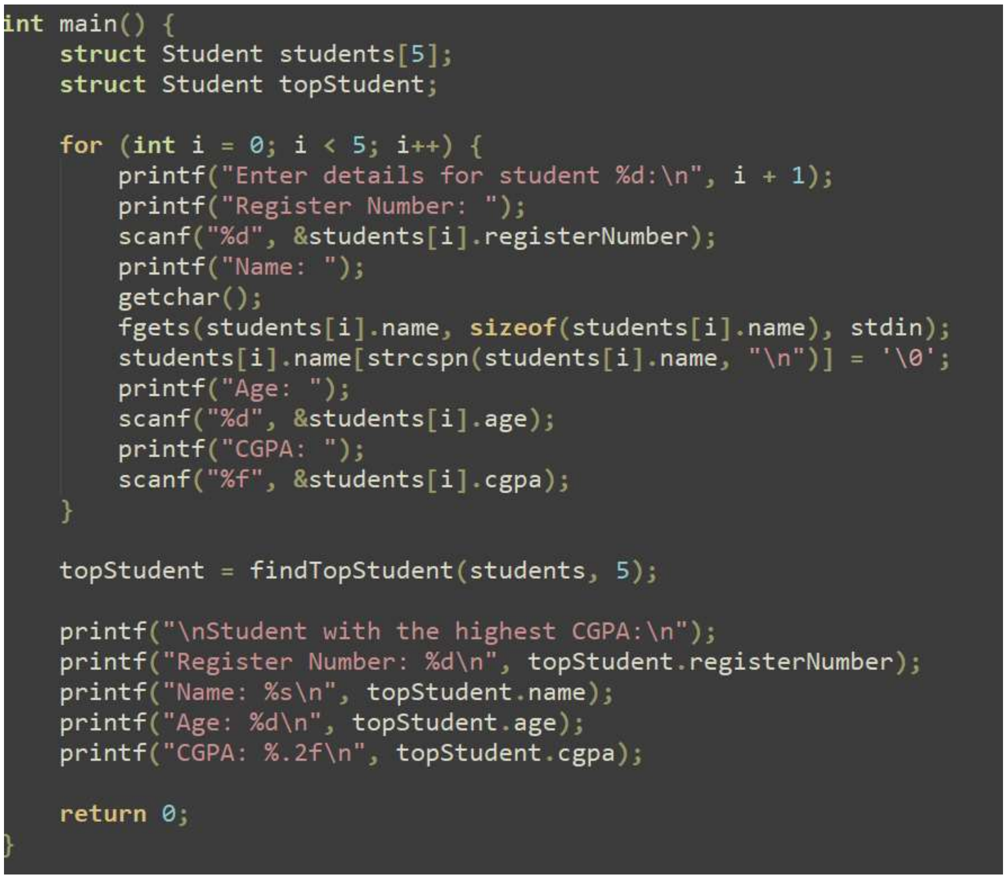
Question 7:

Create a structure for student data base with following members (Register number, Name, Age, CGPA). Write a C program to perform the following operations (i) Get user input for 5 students record (ii) Find the student's name with greatest CGPA.

Answer:

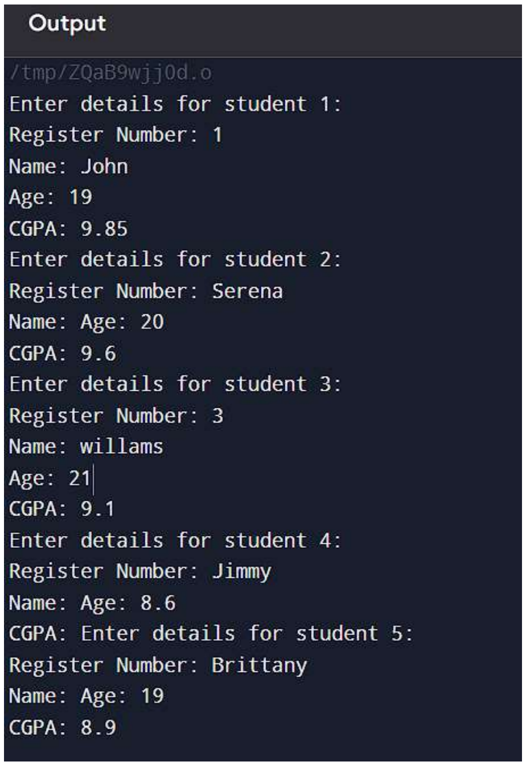


```c
#include <stdio.h>
#include <string.h>

struct Student {
    int registerNumber;
    char name[100];
    int age;
    float cgpa;
};

struct Student findTopStudent(struct Student students[], int size) {
    struct Student topStudent = students[0];

    for (int i = 1; i < size; i++) {
        if (students[i].cgpa > topStudent.cgpa) {
            topStudent = students[i];
        }
    }

    return topStudent;
}
```

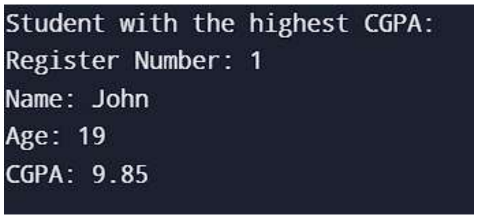


```c
int main() {
    struct Student students[5];
    struct Student topStudent;

    for (int i = 0; i < 5; i++) {
        printf("Enter details for student %d:\n", i + 1);
        printf("Register Number: ");
        scanf("%d", &students[i].registerNumber);
        printf("Name: ");
        getchar();
        fgets(students[i].name, sizeof(students[i].name), stdin);
        students[i].name[strcspn(students[i].name, "\n")] = '\0';
        printf("Age: ");
        scanf("%d", &students[i].age);
        printf("CGPA: ");
        scanf("%f", &students[i].cgpa);
    }

    topStudent = findTopStudent(students, 5);

    printf("\nStudent with the highest CGPA:\n");
    printf("Register Number: %d\n", topStudent.registerNumber);
    printf("Name: %s\n", topStudent.name);
    printf("Age: %d\n", topStudent.age);
    printf("CGPA: %.2f\n", topStudent.cgpa);

    return 0;
}
```

Output:



```
/tmp/ZQaB9wjj0d.o
Enter details for student 1:
Register Number: 1
Name: John
Age: 19
CGPA: 9.85
Enter details for student 2:
Register Number: Serena
Name: Age: 20
CGPA: 9.6
Enter details for student 3:
Register Number: 3
Name: willams
Age: 21
CGPA: 9.1
Enter details for student 4:
Register Number: Jimmy
Name: Age: 8.6
CGPA: Enter details for student 5:
Register Number: Brittany
Name: Age: 19
CGPA: 8.9
```

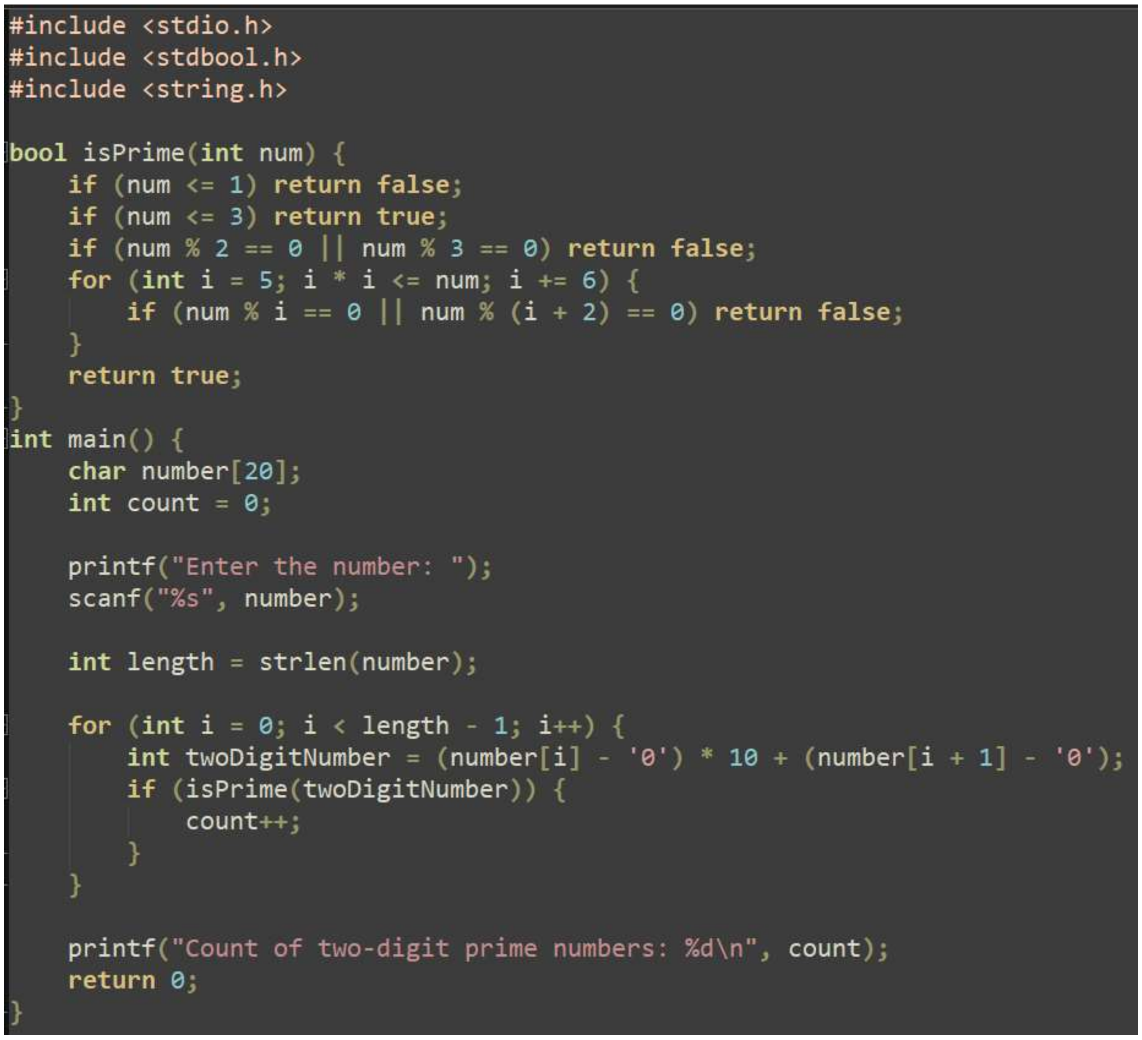


```
Student with the highest CGPA:
Register Number: 1
Name: John
Age: 19
CGPA: 9.85
```

Question 8:

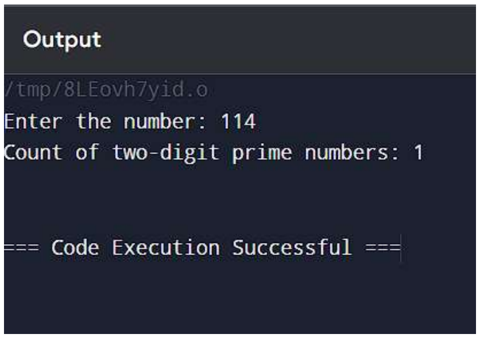
Given a number ‘n’, write an algorithm and the subsequent ‘C’ program to count the number of two-digit prime numbers in it when adjacent digits are taken. For example, if the value of ‘n’ is 114 then the two-digit numbers that can be formed by taking adjacent digits are 11 and 14. 11 is prime but 14 is not. Therefore print 1.

Answer:



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

bool isPrime(int num) {
    if (num <= 1) return false;
    if (num <= 3) return true;
    if (num % 2 == 0 || num % 3 == 0) return false;
    for (int i = 5; i * i <= num; i += 6) {
        if (num % i == 0 || num % (i + 2) == 0) return false;
    }
    return true;
}
int main() {
    char number[20];
    int count = 0;

    printf("Enter the number: ");
    scanf("%s", number);

    int length = strlen(number);

    for (int i = 0; i < length - 1; i++) {
        int twoDigitNumber = (number[i] - '0') * 10 + (number[i + 1] - '0');
        if (isPrime(twoDigitNumber)) {
            count++;
        }
    }

    printf("Count of two-digit prime numbers: %d\n", count);
    return 0;
}
```

Output:



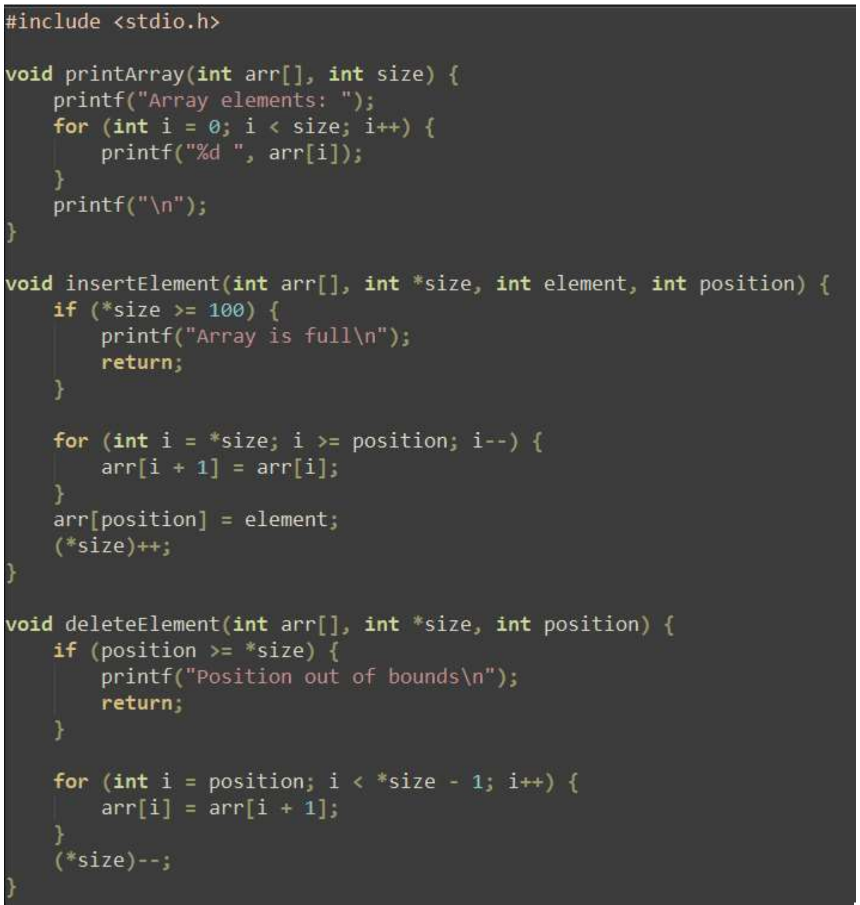
```
/tmp/8LEovh7yid.o
Enter the number: 114
Count of two-digit prime numbers: 1


=== Code Execution Successful ===
```

Question 9:

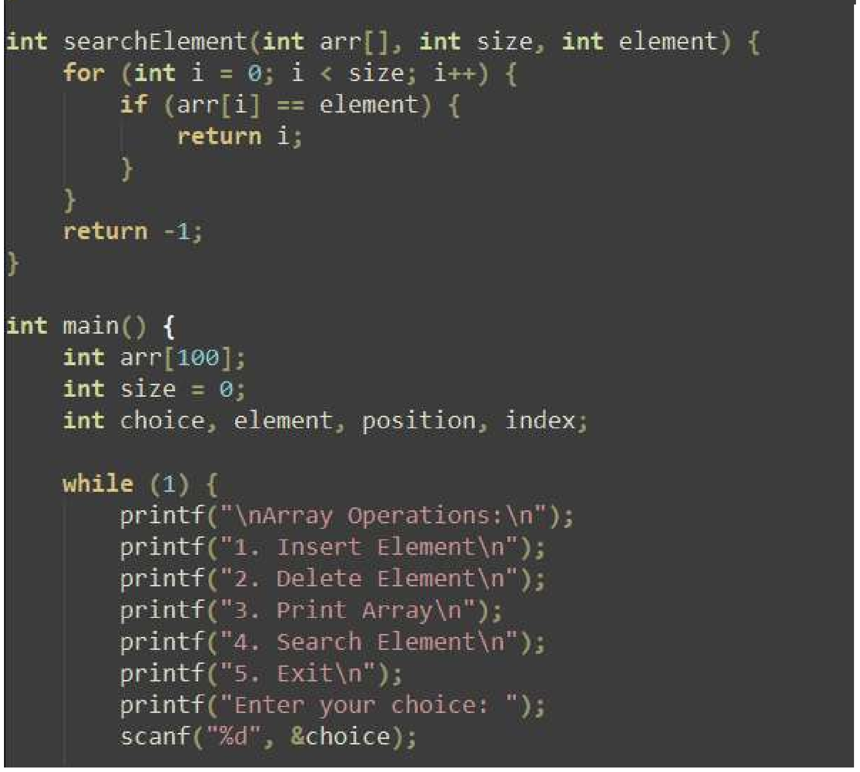
Demonstrate the usage of list and perform all the possible operation using array with suitable examples

.

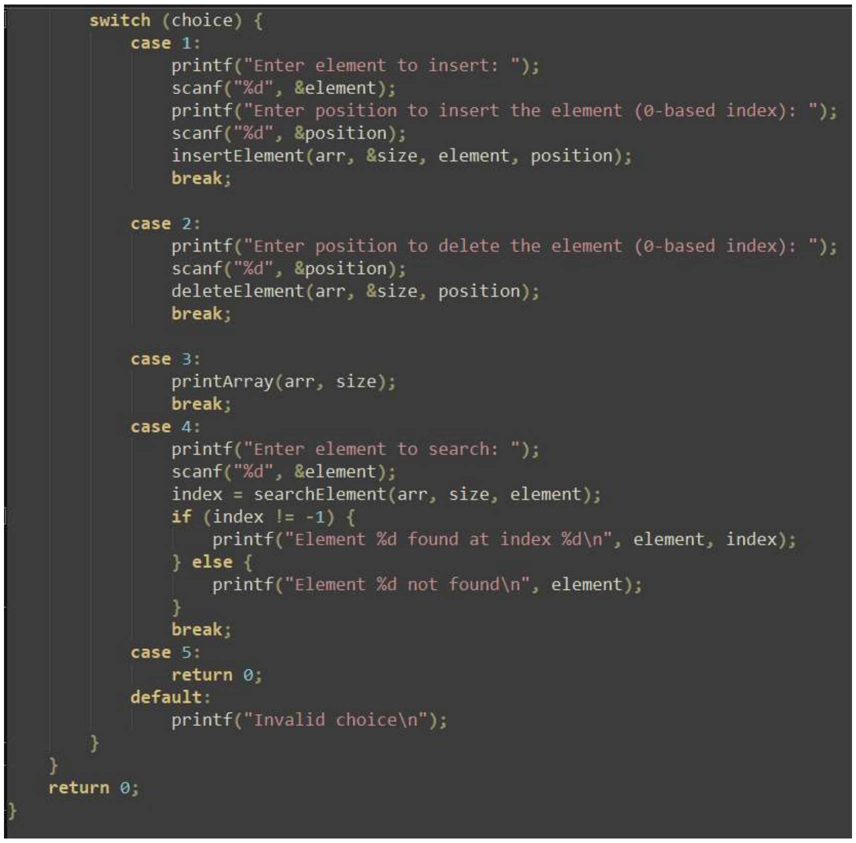
Answer:



```c
#include <stdio.h>

void printArray(int arr[], int size) {
    printf("Array elements: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

void insertElement(int arr[], int *size, int element, int position) {
    if (*size >= 100) {
        printf("Array is full\n");
        return;
    }

    for (int i = *size; i >= position; i--) {
        arr[i + 1] = arr[i];
    }
    arr[position] = element;
    (*size)++;
}

void deleteElement(int arr[], int *size, int position) {
    if (position >= *size) {
        printf("Position out of bounds\n");
        return;
    }

    for (int i = position; i < *size - 1; i++) {
        arr[i] = arr[i + 1];
    }
    (*size)--;
}
```

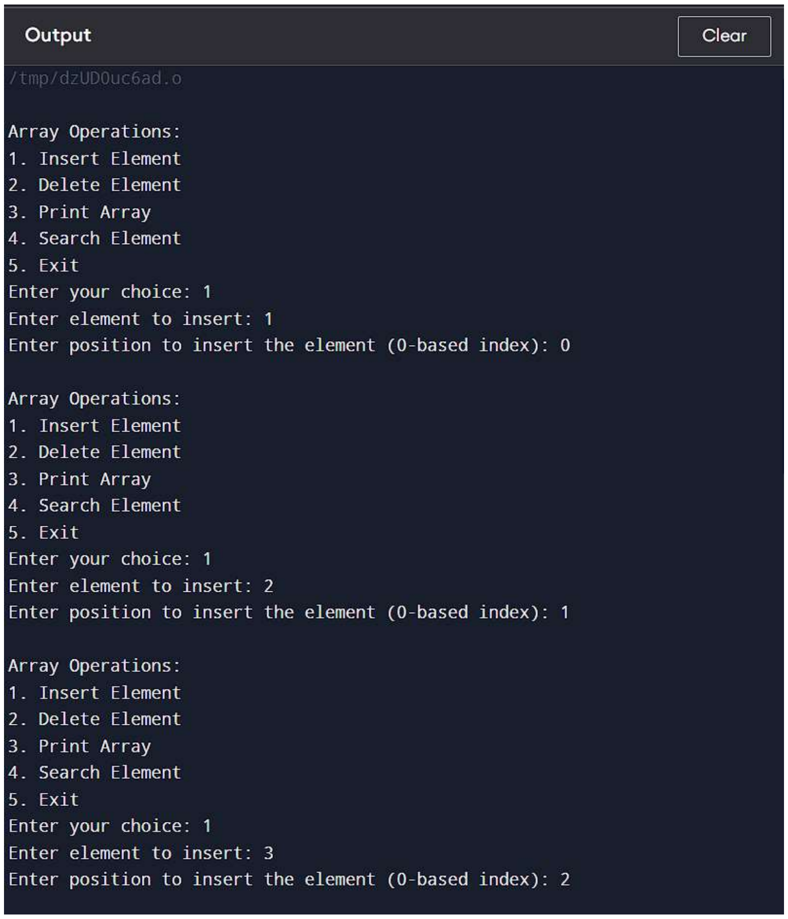


```c
int searchElement(int arr[], int size, int element) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == element) {
            return i;
        }
    }
    return -1;
}

int main() {
    int arr[100];
    int size = 0;
    int choice, element, position, index;

    while (1) {
        printf("\nArray Operations:\n");
        printf("1. Insert Element\n");
        printf("2. Delete Element\n");
        printf("3. Print Array\n");
        printf("4. Search Element\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

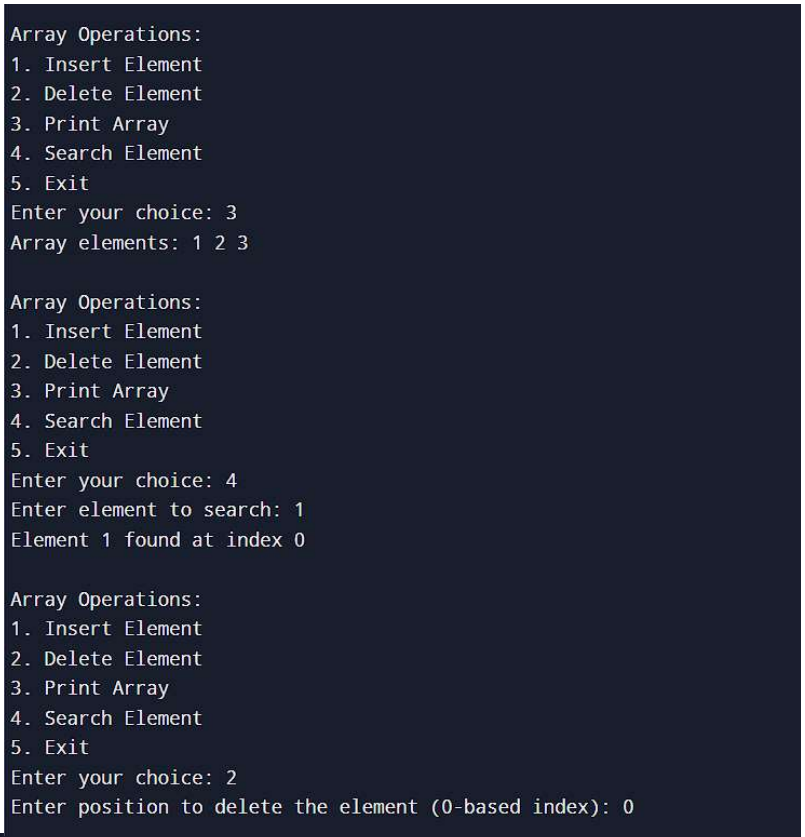


```
        switch (choice) {
            case 1:
                printf("Enter element to insert: ");
                scanf("%d", &element);
                printf("Enter position to insert the element (0-based index): ");
                scanf("%d", &position);
                insertElement(arr, &size, element, position);
                break;

            case 2:
                printf("Enter position to delete the element (0-based index): ");
                scanf("%d", &position);
                deleteElement(arr, &size, position);
                break;

            case 3:
                printArray(arr, size);
                break;
            case 4:
                printf("Enter element to search: ");
                scanf("%d", &element);
                index = searchElement(arr, size, element);
                if (index != -1) {
                    printf("Element %d found at index %d\n", element, index);
                } else {
                    printf("Element %d not found\n", element);
                }
                break;
            case 5:
                return 0;
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

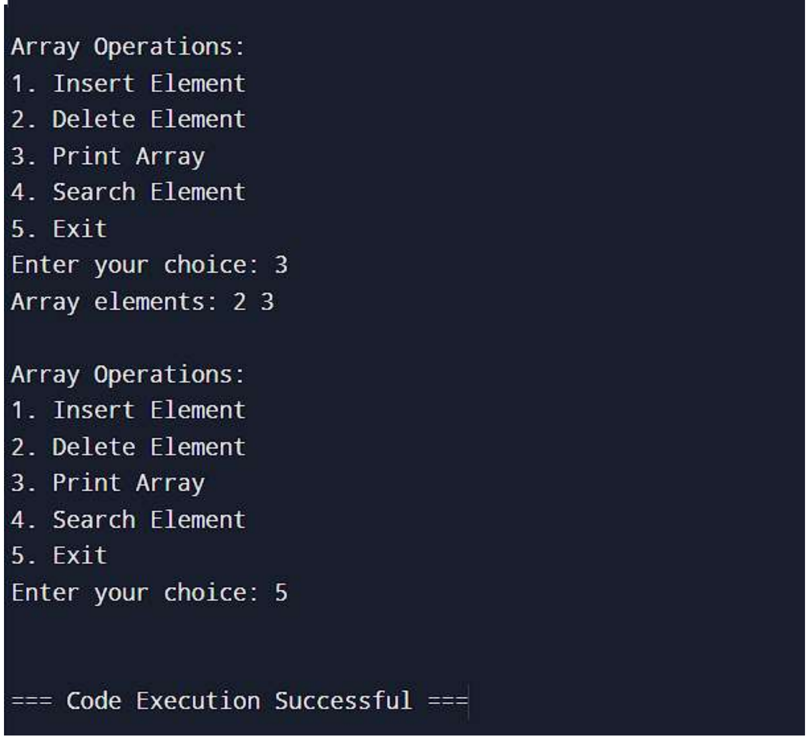
Output:



```
/tmp/dzUDOuc6ad.o

Array Operations:
1. Insert Element
2. Delete Element
3. Print Array
4. Search Element
5. Exit
Enter your choice: 1
Enter element to insert: 1
Enter position to insert the element (0-based index): 0

Array Operations:
1. Insert Element
2. Delete Element
3. Print Array
4. Search Element
5. Exit
Enter your choice: 1
Enter element to insert: 2
Enter position to insert the element (0-based index): 1

Array Operations:
1. Insert Element
2. Delete Element
3. Print Array
4. Search Element
5. Exit
Enter your choice: 1
Enter element to insert: 3
Enter position to insert the element (0-based index): 2
```



```
Array Operations:
1. Insert Element
2. Delete Element
3. Print Array
4. Search Element
5. Exit
Enter your choice: 3
Array elements: 1 2 3

Array Operations:
1. Insert Element
2. Delete Element
3. Print Array
4. Search Element
5. Exit
Enter your choice: 4
Enter element to search: 1
Element 1 found at index 0

Array Operations:
1. Insert Element
2. Delete Element
3. Print Array
4. Search Element
5. Exit
Enter your choice: 2
Enter position to delete the element (0-based index): 0
```



```
Array Operations:
1. Insert Element
2. Delete Element
3. Print Array
4. Search Element
5. Exit
Enter your choice: 3
Array elements: 2 3

Array Operations:
1. Insert Element
2. Delete Element
3. Print Array
4. Search Element
5. Exit
Enter your choice: 5


=== Code Execution Successful ===
```