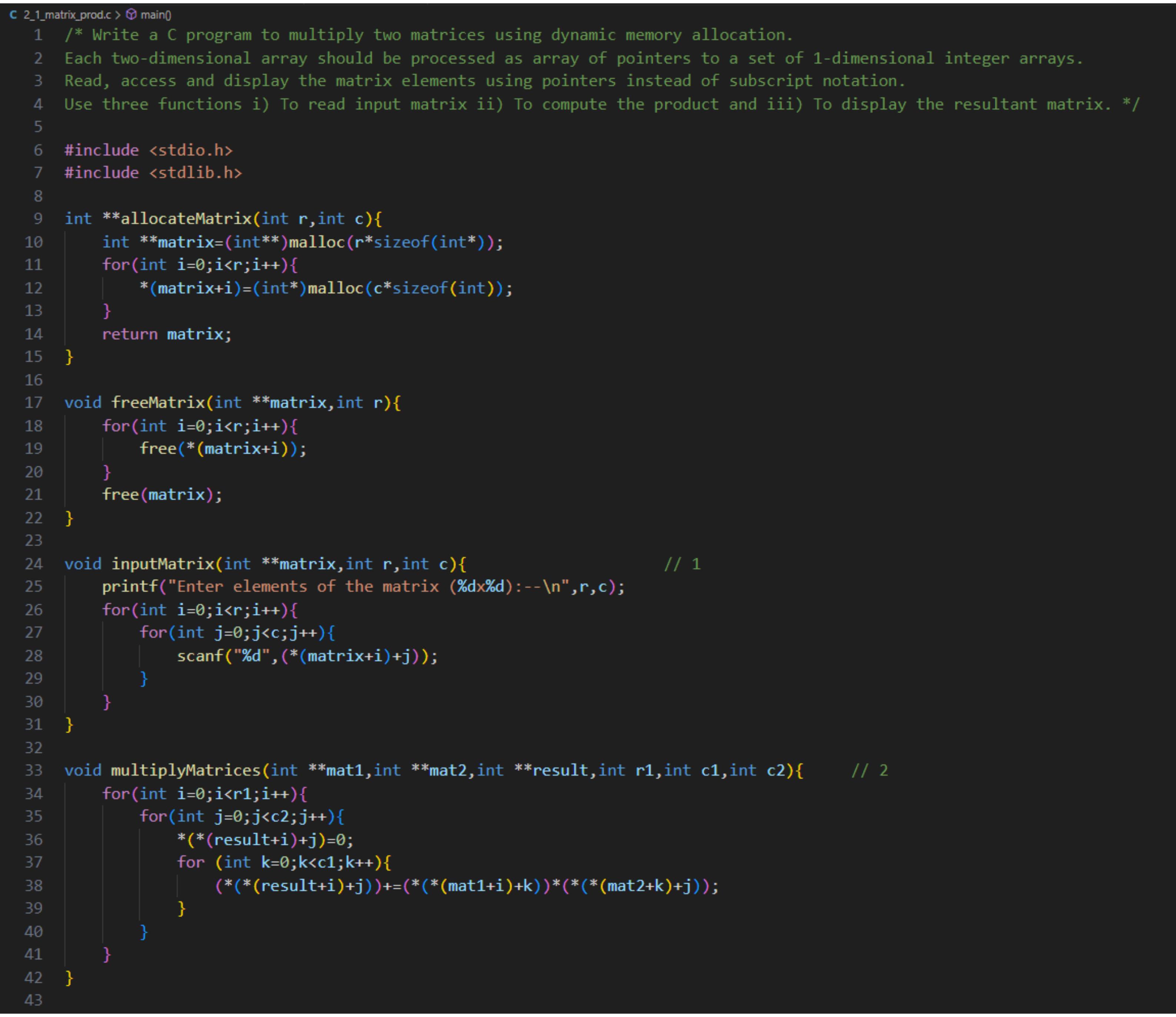
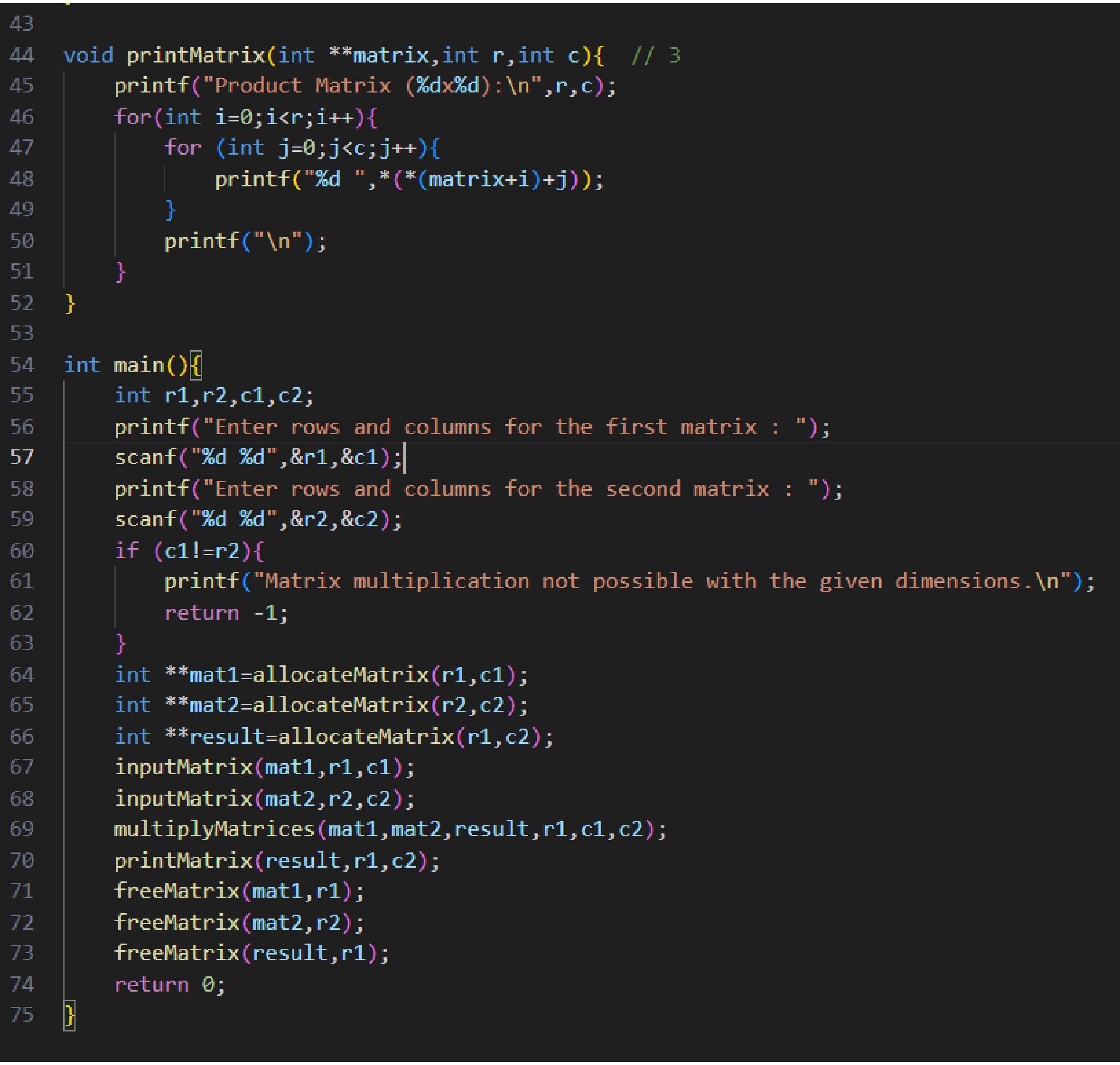
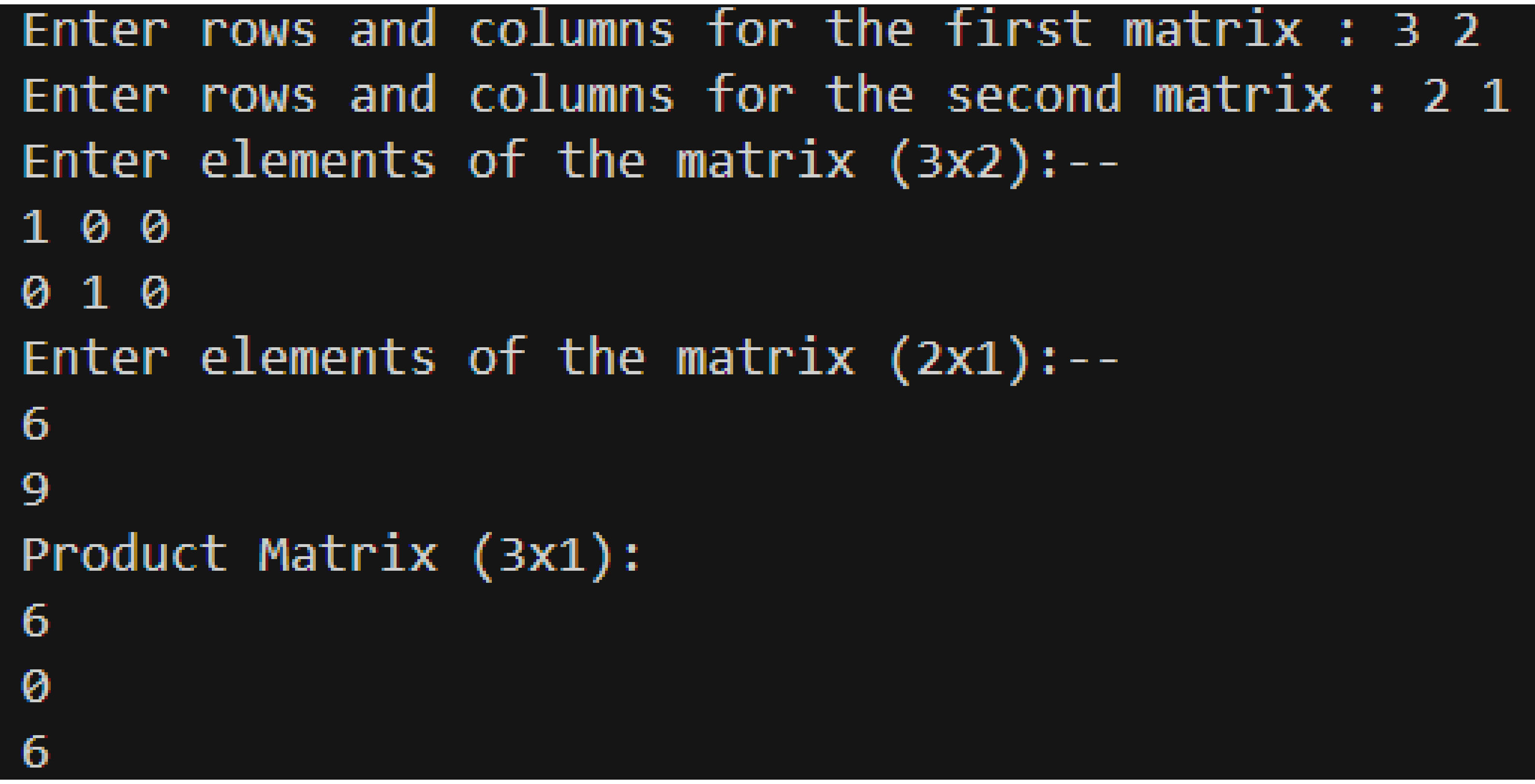


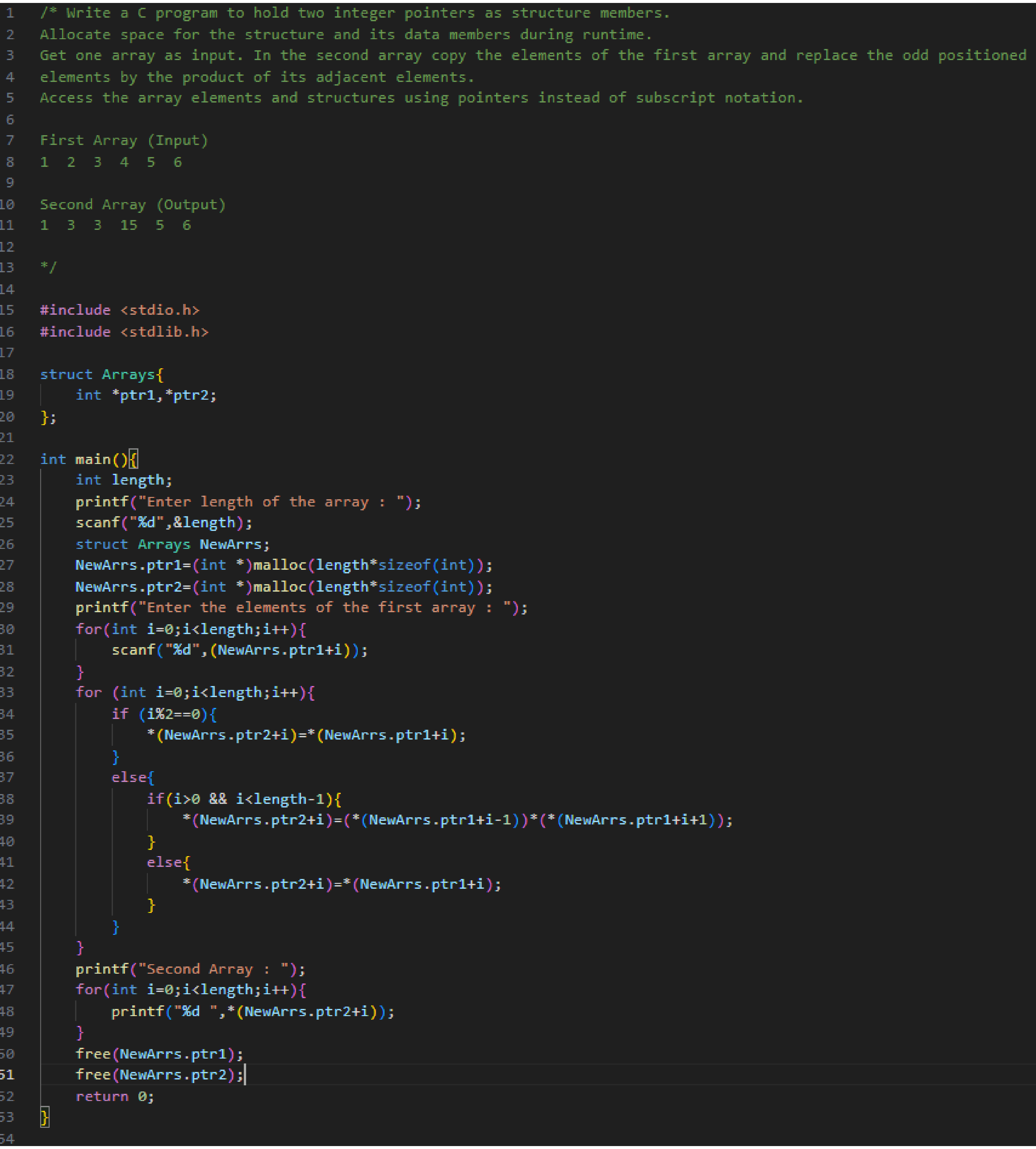
```c
/* Write a C program to multiply two matrices using dynamic memory allocation.
Each two-dimensional array should be processed as array of pointers to a set of 1-dimensional integer arrays.
Read, access and display the matrix elements using pointers instead of subscript notation.
Use three functions i) To read input matrix ii) To compute the product and iii) To display the resultant matrix. */

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

int **allocateMatrix(int r,int c){
    int **matrix=(int**)malloc(r*sizeof(int*));
    for(int i=0;i<r;i++){
        *(matrix+i)=(int*)malloc(c*sizeof(int));
    }
    return matrix;
}

void freeMatrix(int **matrix,int r){
    for(int i=0;i<r;i++){
        free(*(matrix+i));
    }
    free(matrix);
}

void inputMatrix(int **matrix,int r,int c){                     // 1
    printf("Enter elements of the matrix (%dx%d):--\n",r,c);
    for(int i=0;i<r;i++){
        for(int j=0;j<c;j++){
            scanf("%d",(*(matrix+i)+j));
        }
    }
}

void multiplyMatrices(int **mat1,int **mat2,int **result,int r1,int c1,int c2){     // 2
    for(int i=0;i<r1;i++){
        for(int j=0;j<c2;j++){
            *(*(result+i)+j)=0;
            for (int k=0;k<c1;k++){
                (*(*(result+i)+j))+=(*(*(mat1+i)+k))*(*(*(mat2+k)+j));
            }
        }
    }
}

```



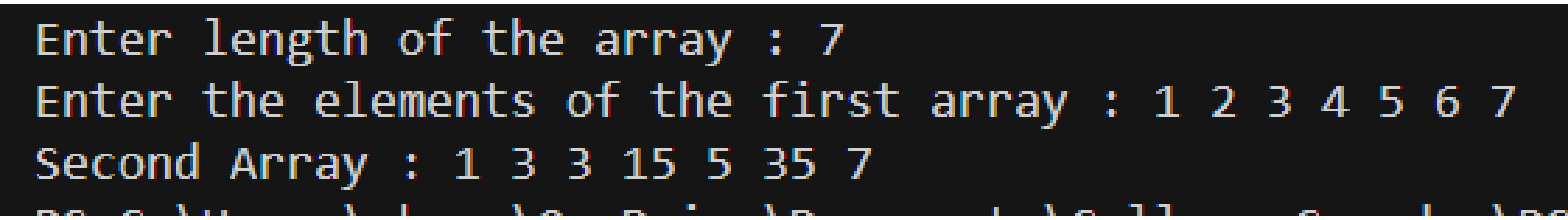
```c

void printMatrix(int **matrix,int r,int c){  // 3
    printf("Product Matrix (%dx%d):\n",r,c);
    for(int i=0;i<r;i++){
        for (int j=0;j<c;j++){
            printf("%d ",*(*(matrix+i)+j));
        }
        printf("\n");
    }
}

int main(){
    int r1,r2,c1,c2;
    printf("Enter rows and columns for the first matrix : ");
    scanf("%d %d",&r1,&c1);
    printf("Enter rows and columns for the second matrix : ");
    scanf("%d %d",&r2,&c2);
    if (c1!=r2){
        printf("Matrix multiplication not possible with the given dimensions.\n");
        return -1;
    }
    int **mat1=allocateMatrix(r1,c1);
    int **mat2=allocateMatrix(r2,c2);
    int **result=allocateMatrix(r1,c2);
    inputMatrix(mat1,r1,c1);
    inputMatrix(mat2,r2,c2);
    multiplyMatrices(mat1,mat2,result,r1,c1,c2);
    printMatrix(result,r1,c2);
    freeMatrix(mat1,r1);
    freeMatrix(mat2,r2);
    freeMatrix(result,r1);
    return 0;
}
```



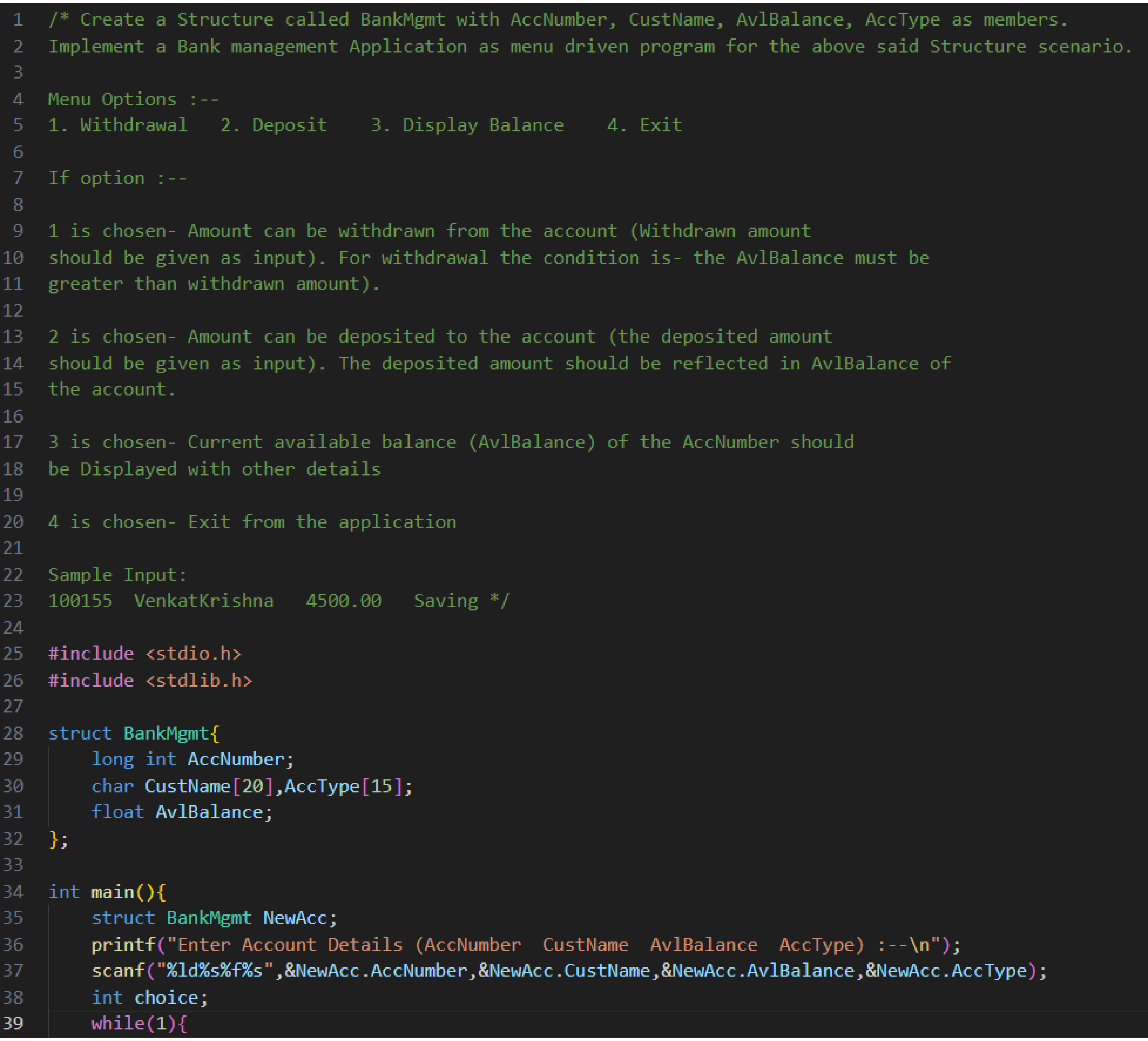
```
Enter rows and columns for the first matrix : 3 2
Enter rows and columns for the second matrix : 2 1
Enter elements of the matrix (3x2):--
1 0 0
0 1 0
Enter elements of the matrix (2x1):--
6
9
Product Matrix (3x1):
6
0
6
```



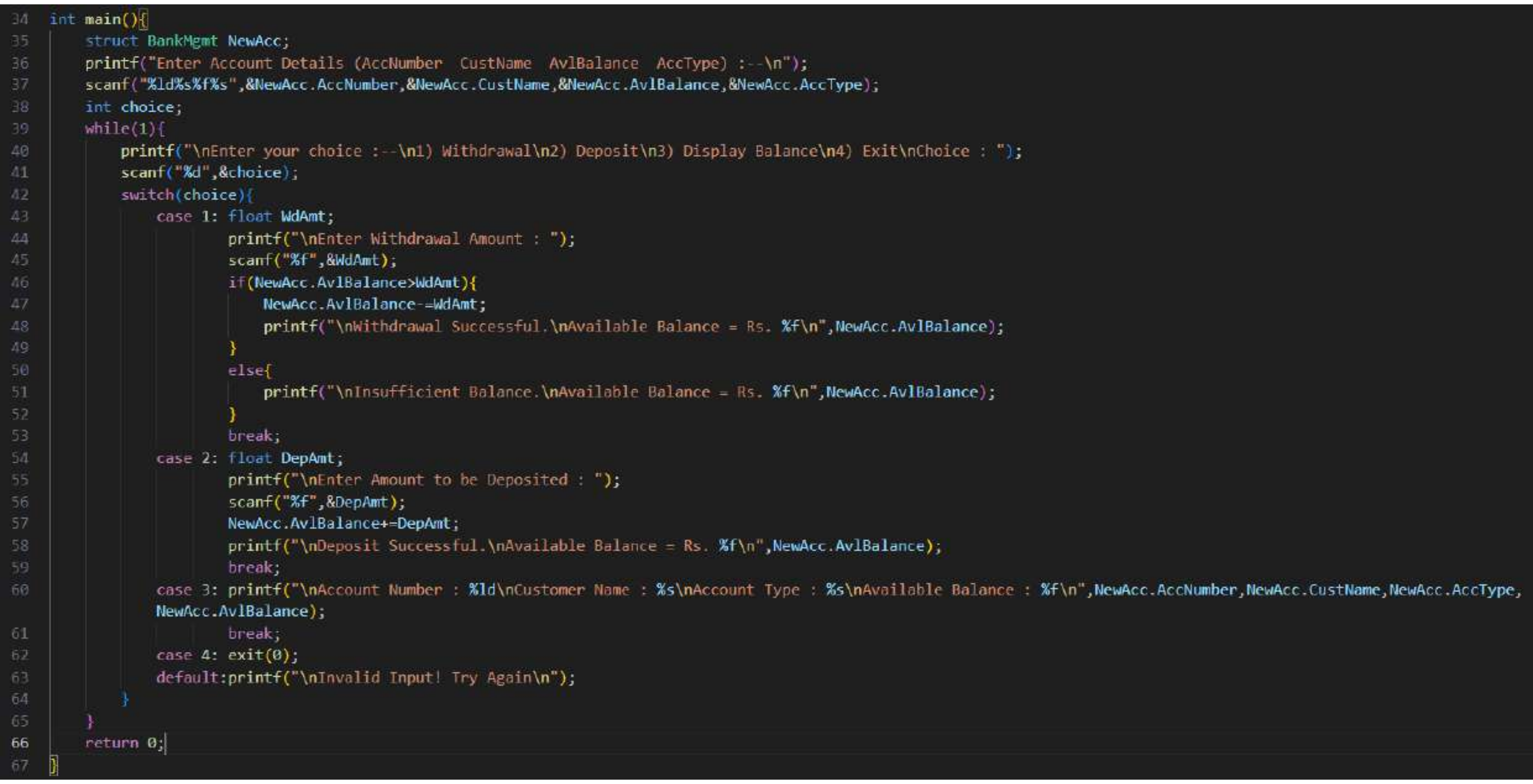
```c
/* Write a C program to hold two integer pointers as structure members.
Allocate space for the structure and its data members during runtime.
Get one array as input. In the second array copy the elements of the first array and replace the odd positioned
elements by the product of its adjacent elements.
Access the array elements and structures using pointers instead of subscript notation.

First Array (Input)
1  2  3  4  5  6

Second Array (Output)
1  3  3  15  5  6

*/

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

struct Arrays{
    int *ptr1,*ptr2;
};

int main(){
    int length;
    printf("Enter length of the array : ");
    scanf("%d",&length);
    struct Arrays NewArrs;
    NewArrs.ptr1=(int *)malloc(length*sizeof(int));
    NewArrs.ptr2=(int *)malloc(length*sizeof(int));
    printf("Enter the elements of the first array : ");
    for(int i=0;i<length;i++){
        scanf("%d",(NewArrs.ptr1+i));
    }
    for (int i=0;i<length;i++){
        if (i%2==0){
            *(NewArrs.ptr2+i)=*(NewArrs.ptr1+i);
        }
        else{
            if(i>0 && i<length-1){
                *(NewArrs.ptr2+i)=(*(NewArrs.ptr1+i-1))*(*(NewArrs.ptr1+i+1));
            }
            else{
                *(NewArrs.ptr2+i)=*(NewArrs.ptr1+i);
            }
        }
    }
    printf("Second Array : ");
    for(int i=0;i<length;i++){
        printf("%d ",*(NewArrs.ptr2+i));
    }
    free(NewArrs.ptr1);
    free(NewArrs.ptr2);
    return 0;
}
```



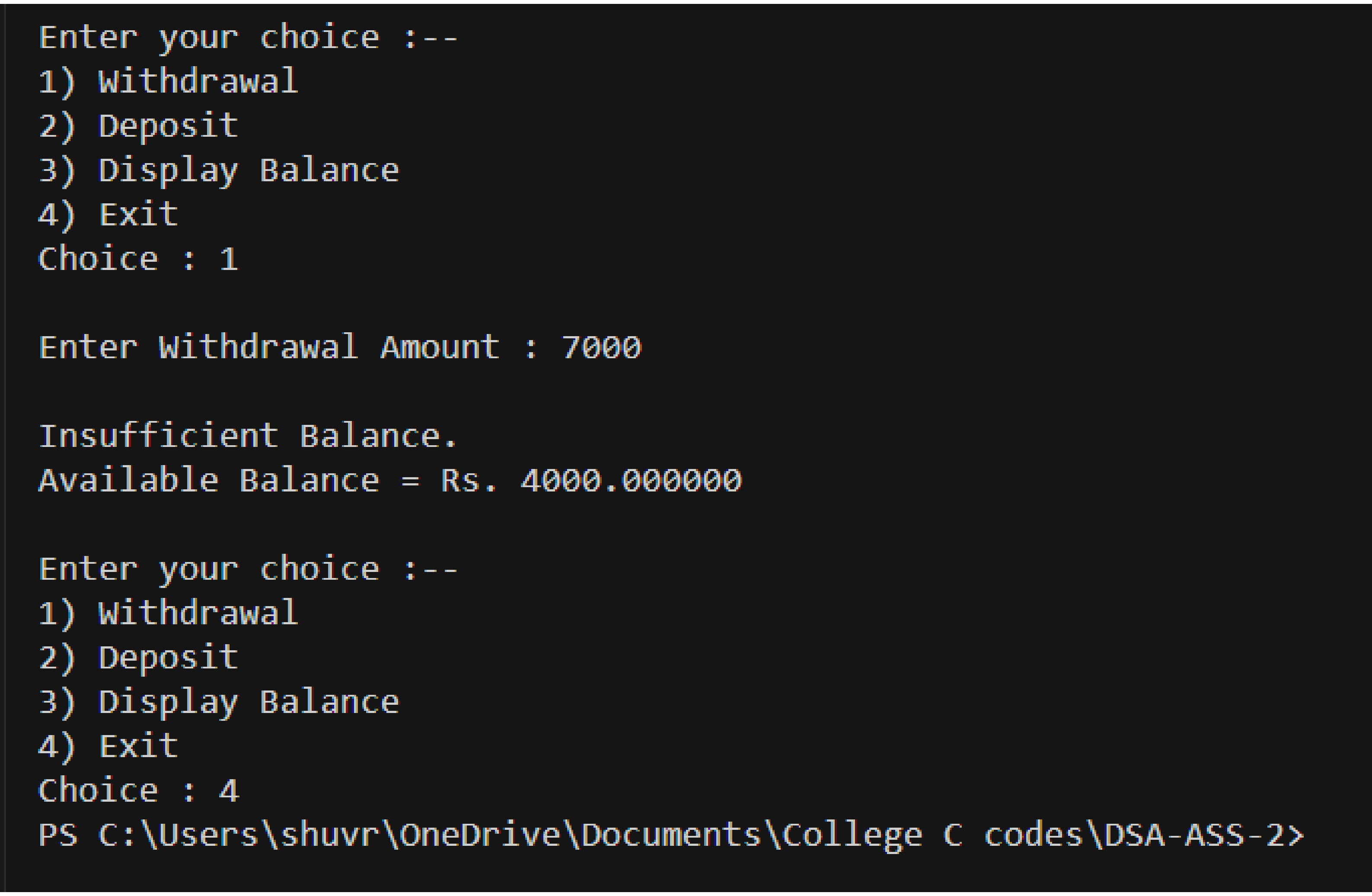
```
Enter length of the array : 7
Enter the elements of the first array : 1 2 3 4 5 6 7
Second Array : 1 3 3 15 5 35 7
```



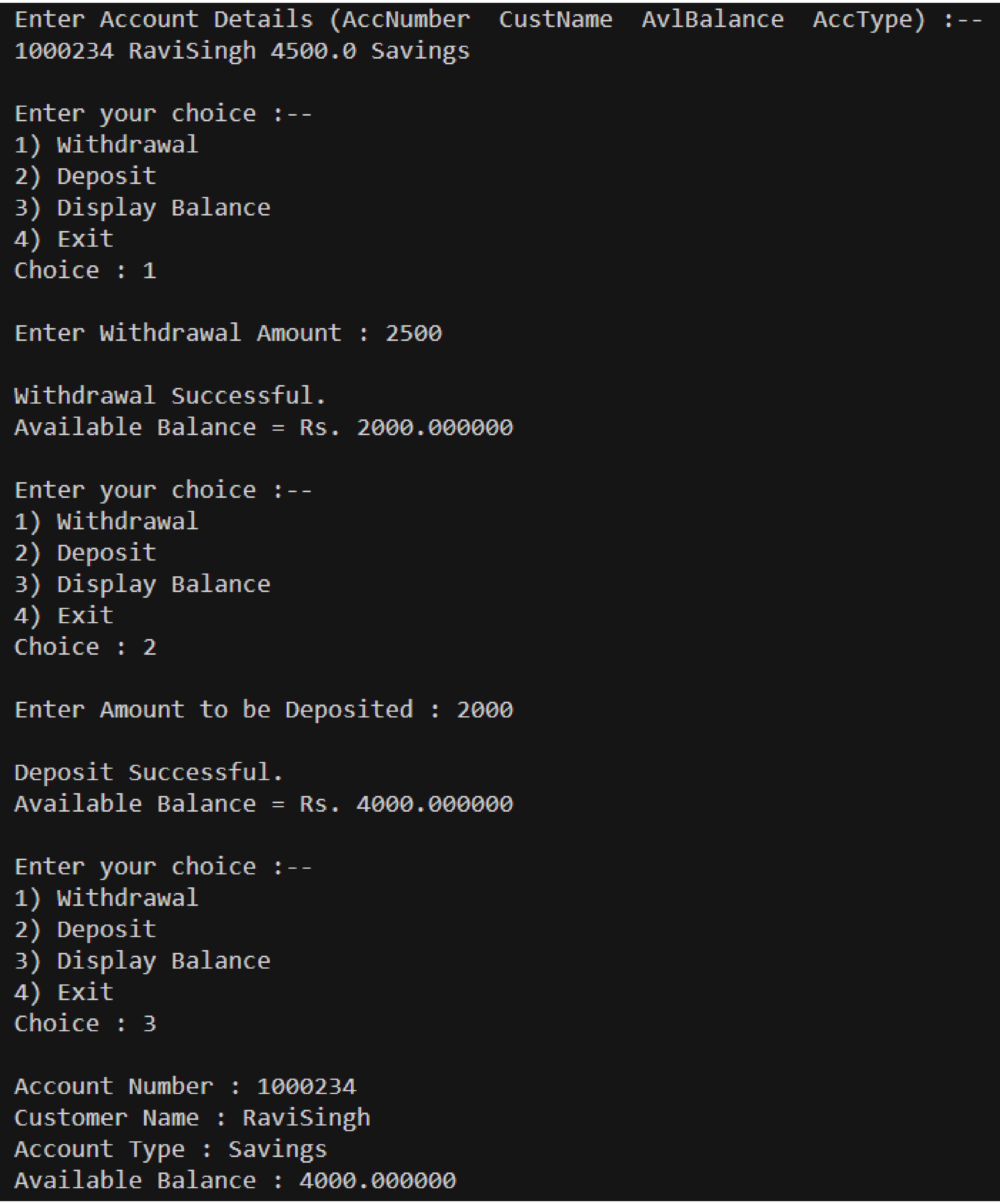
```c
/* Create a Structure called BankMgmt with AccNumber, CustName, AvlBalance, AccType as members.
Implement a Bank management Application as menu driven program for the above said Structure scenario.

Menu Options :--
1. Withdrawal   2. Deposit    3. Display Balance    4. Exit

If option :--

1 is chosen- Amount can be withdrawn from the account (Withdrawn amount
should be given as input). For withdrawal the condition is- the AvlBalance must be
greater than withdrawn amount).

2 is chosen- Amount can be deposited to the account (the deposited amount
should be given as input). The deposited amount should be reflected in AvlBalance of
the account.

3 is chosen- Current available balance (AvlBalance) of the AccNumber should
be Displayed with other details

4 is chosen- Exit from the application

Sample Input:
100155  VenkatKrishna   4500.00   Saving */

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

struct BankMgmt{
    long int AccNumber;
    char CustName[20],AccType[15];
    float AvlBalance;
};

int main(){
    struct BankMgmt NewAcc;
    printf("Enter Account Details (AccNumber  CustName  AvlBalance  AccType) :--\n");
    scanf("%ld%s%f%s",&NewAcc.AccNumber,&NewAcc.CustName,&NewAcc.AvlBalance,&NewAcc.AccType);
    int choice;
    while(1){
```



```c
int main(){
    struct BankMgmt NewAcc;
    printf("Enter Account Details (AccNumber  CustName  AvlBalance  AccType) :--\n");
    scanf("%ld%s%f%s",&NewAcc.AccNumber,&NewAcc.CustName,&NewAcc.AvlBalance,&NewAcc.AccType);
    int choice;
    while(1){
        printf("\nEnter your choice :--\n1) Withdrawal\n2) Deposit\n3) Display Balance\n4) Exit\nChoice : ");
        scanf("%d",&choice);
        switch(choice){
            case 1: float WdAmt;
                    printf("\nEnter Withdrawal Amount : ");
                    scanf("%f",&WdAmt);
                    if(NewAcc.AvlBalance>WdAmt){
                        NewAcc.AvlBalance-=WdAmt;
                        printf("\nWithdrawal Successful.\nAvailable Balance = Rs. %f\n",NewAcc.AvlBalance);
                    }
                    else{
                        printf("\nInsufficient Balance.\nAvailable Balance = Rs. %f\n",NewAcc.AvlBalance);
                    }
                    break;
            case 2: float DepAmt;
                    printf("\nEnter Amount to be Deposited : ");
                    scanf("%f",&DepAmt);
                    NewAcc.AvlBalance+=DepAmt;
                    printf("\nDeposit Successful.\nAvailable Balance = Rs. %f\n",NewAcc.AvlBalance);
                    break;
            case 3: printf("\nAccount Number : %ld\nCustomer Name : %s\nAccount Type : %s\nAvailable Balance : %f\n",NewAcc.AccNumber,NewAcc.CustName,NewAcc.AccType,
            NewAcc.AvlBalance);
                    break;
            case 4: exit(0);
            default:printf("\nInvalid Input! Try Again\n");
        }
    }
    return 0;
}
```



```
Enter your choice :--
1) Withdrawal
2) Deposit
3) Display Balance
4) Exit
Choice : 1

Enter Withdrawal Amount : 7000

Insufficient Balance.
Available Balance = Rs. 4000.000000

Enter your choice :--
1) Withdrawal
2) Deposit
3) Display Balance
4) Exit
Choice : 4
PS C:\Users\shuvr\OneDrive\Documents\College C codes\DSA-ASS-2>
```



```
Enter Account Details (AccNumber  CustName  AvlBalance  AccType) :--
1000234 RaviSingh 4500.0 Savings

Enter your choice :--
1) Withdrawal
2) Deposit
3) Display Balance
4) Exit
Choice : 1

Enter Withdrawal Amount : 2500

Withdrawal Successful.
Available Balance = Rs. 2000.000000

Enter your choice :--
1) Withdrawal
2) Deposit
3) Display Balance
4) Exit
Choice : 2

Enter Amount to be Deposited : 2000

Deposit Successful.
Available Balance = Rs. 4000.000000

Enter your choice :--
1) Withdrawal
2) Deposit
3) Display Balance
4) Exit
Choice : 3

Account Number : 1000234
Customer Name : RaviSingh
Account Type : Savings
Available Balance : 4000.000000
```