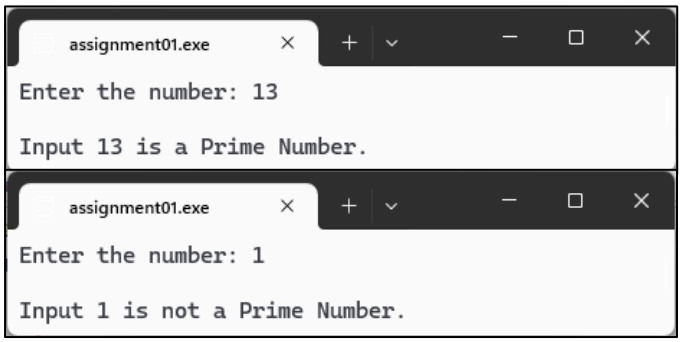
Assignment 1:

Write a C program that includes a user-defined function named isPrime with the signature **int isPrime(int num);** The function should take an integer as a parameter and return 1 if the number is prime and 0 otherwise.

Source Code:

```
#include <stdio.h>
#include <math.h>

int isPrime(int);

int main()
{
    int n;
    printf("\n\nEnter the number: ");
    scanf("%d", &n);
    if (isPrime(n))
    {
        printf("\n\nInput %d is a Prime Number.\n", n);
    }
    else
    {
        printf("\n\nInput %d is not a Prime Number.\n", n);
    }
    return 0;
}

int isPrime(int n)
{
    if (n <= 1)
        return 0;
    if (n == 2)
        return 1;
    if (n % 2 == 0)
        return 0;
    int temp = (int)sqrt(n);
    int i;
    for (i = 3; i <= temp; i += 2)
    {
        if (n % i == 0)
        {
            return 0;
        }
    }
    return 1;
}
```

Output:



Assignment 2:

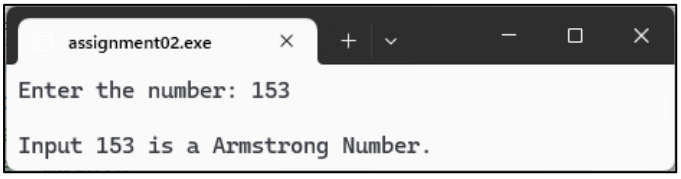
Write a C program that includes a user-defined function named isArmstrong with the signature **int isArmstrong(int num);** . An Armstrong number is a number that is equal to the sum of its own digits each raised to the power of the number of digits. For example, 153 is an Armstrong number because 1^3 + 5^3 + 3^3 = 153

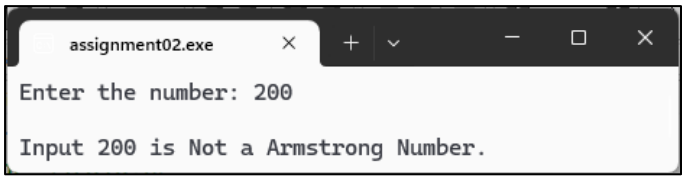
Source Code:

```
#include <stdio.h>
#include <math.h>

int isArmstrong(int);
int count(int);

int main()
{
    int n;
    printf("Enter the number: ");
    scanf("%d", &n);

    if (isArmstrong(n))
    {
        printf("\nInput %d is a Armstrong Number.", n);
    }
    else
    {
        printf("\nInput %d is Not a Armstrong Number.", n);
    }

    return 0;
}

int count(int n)
{
    int count = 0;
    while (n > 0)
    {
        count++;
        n = n / 10;
    }
    return count;
}

int isArmstrong(int n)
{
    if (n < 0)
        return 0;
    if (n == 0)
        return 1;

    int power = count(n);
    int temp = n;
    int checker = 0;

    while (temp > 0)
```

```
    {
        int digit = temp % 10;
        checker = checker + (int)round(pow(digit, power));
        temp = temp / 10;
    }
    return n == checker;
}
```

Output:





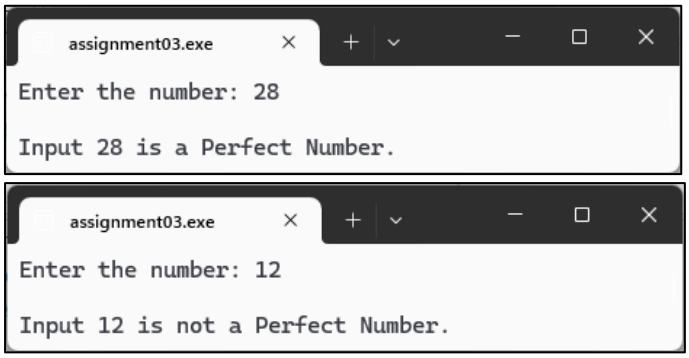
Assignment 3:

Write a C program that includes a user-defined function named isPerfect with the signature **int isPerfect(int num);** . A perfect number is a positive integer that is equal to the sum of its proper divisors, excluding itself. For example, 28 is a perfect number because the sum of its divisors (1, 2, 4, 7, 14) equals 28.

Source Code:

```
#include <stdio.h>

int isPerfect(int);

int main()
{
    int n;
    printf("Enter the number: ");
    scanf("%d", &n);
    if (isPerfect(n))
    {
        printf("\nInput %d is a Perfect Number.", n);
    }
    else
    {
        printf("\n\nInput %d is not a Perfect Number.", n);
    }
    return 0;
}

int isPerfect(int n)
{
    if (n <= 1)
        return 0;
    int temp = 1;
    int i;
    for (i = 2; i <= n / 2; i++)
    {
        if (n % i == 0)
        {
            temp += i;
        }
    }
    return temp == n;
}
```

Output:



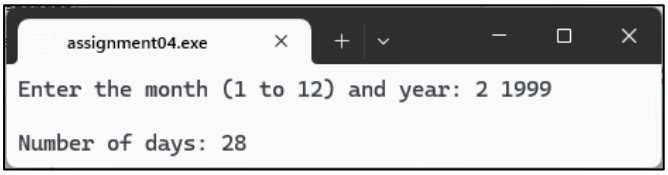
Assignment 4:

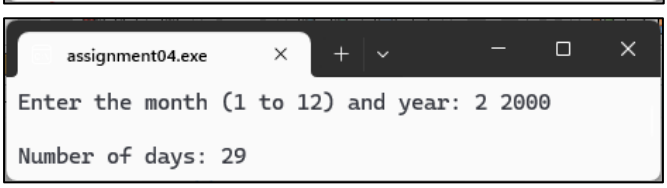
Write a C program that takes an integer input representing a month (1 to 12) and a year. Use a switch statement to display the number of days in that month, considering leap years.

Source Code:

```
#include <stdio.h>
int main()
{
    int month, year, days;
    printf("Enter the month (1 to 12) and year: ");
    scanf("%d %d", &month, &year);
    switch (month)
    {
    case 1:  // jan
    case 3:  // mar
    case 5:  // may
    case 7:  // jul
    case 8:  // aug
    case 10: // oct
    case 12: // dec
        days = 31;
        break;
    case 4:  // apr
    case 6:  // jun
    case 9:  // sep
    case 11: // nov
        days = 30;
        break;
    case 2: // feb
        if ((year % 400 == 0) || ((year % 4 == 0) && (year % 100 != 0)))
        {
            days = 29;
        }
        else
        {
            days = 28;
        }
        break;
    default:
        printf("\nYou entered something wrong.");
        return 1;
    }
    printf("\nNumber of days: %d", days);
    return 0;
}
```

Output:





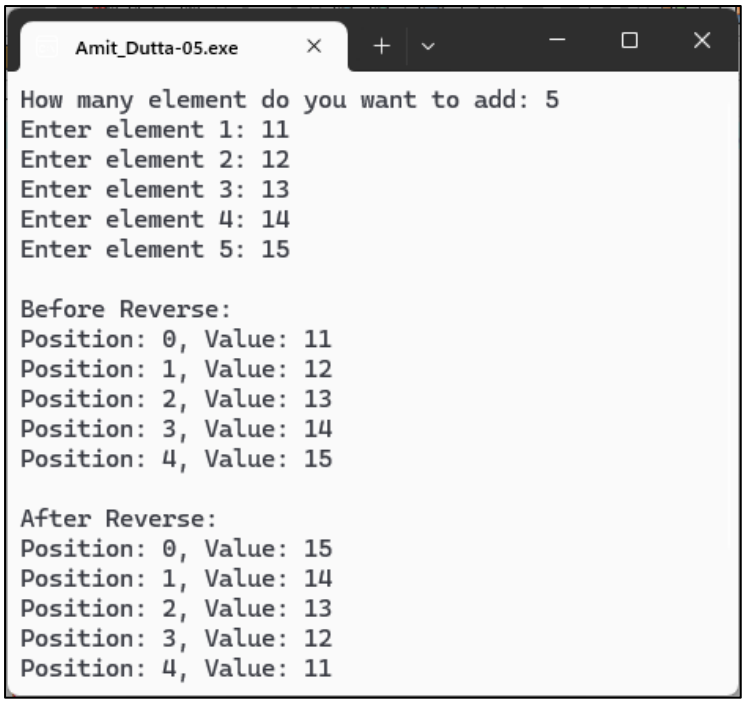
Assignment 5:

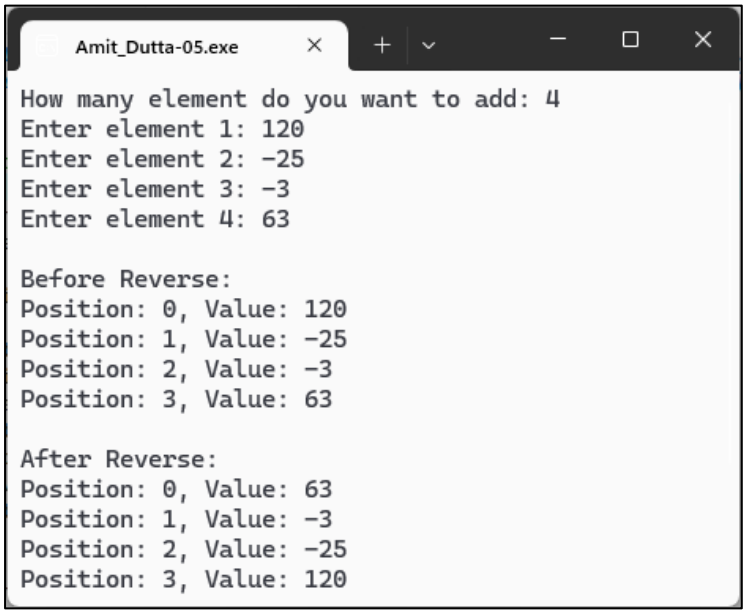
Write a C program that defines an array of integers, and includes a user-defined function named reverseArray with the signature **void reverseArray(int arr[], int size);**. The function should reverse the elements of the array.

Source Code:

```
#include <stdio.h>

void inputArray(int[], int);
void reverseArray(int[], int);

int main()
{
    int size;
    printf("How many element do you want to add: ");
    scanf("%d", &size);
    int arr[size];
    inputArray(arr, size);
    reverseArray(arr, size);
    return 0;
}

void inputArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
    {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &arr[i]);
    }
}

void reverseArray(int arr[], int size)
{
    int i, j, temp;
    printf("\nBefore Reverse: \n");
    for (i = 0; i < size; i++)
    {
        printf("Position: %d, Value: %d\n", i, arr[i]);
    }

    for (i = 0, j = size - 1; i < j; i++, j--)
    {
        temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    printf("\nAfter Reverse: \n");
    for (i = 0; i < size; i++)
    {
        printf("Position: %d, Value: %d\n", i, arr[i]);
    }
}
```

Output:


Amit_Dutta-05.exe
How many element do you want to add: 5
Enter element 1: 11
Enter element 2: 12
Enter element 3: 13
Enter element 4: 14
Enter element 5: 15
Before Reverse:
Position: 0, Value: 11
Position: 1, Value: 12
Position: 2, Value: 13
Position: 3, Value: 14
Position: 4, Value: 15
After Reverse:
Position: 0, Value: 15
Position: 1, Value: 14
Position: 2, Value: 13
Position: 3, Value: 12
Position: 4, Value: 11


Amit_Dutta-05.exe
How many element do you want to add: 4
Enter element 1: 120
Enter element 2: -25
Enter element 3: -3
Enter element 4: 63
Before Reverse:
Position: 0, Value: 120
Position: 1, Value: -25
Position: 2, Value: -3
Position: 3, Value: 63
After Reverse:
Position: 0, Value: 63
Position: 1, Value: -3
Position: 2, Value: -25
Position: 3, Value: 120

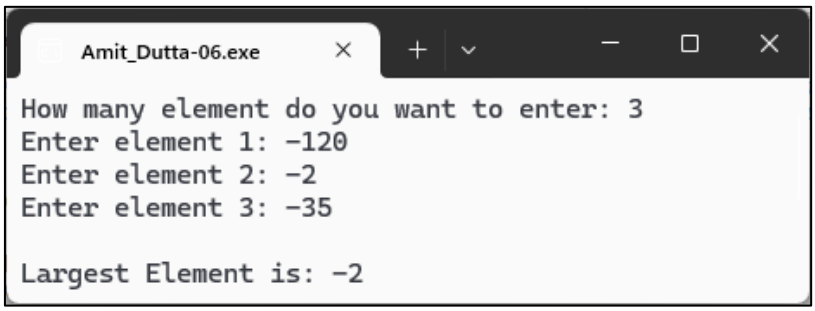
Assignment 6:

Write a C program that includes a user-defined function named findLargest with the signature **int findLargest(int arr[], int size);**. The function should take an array of integers and its size, and return the largest element in the array.

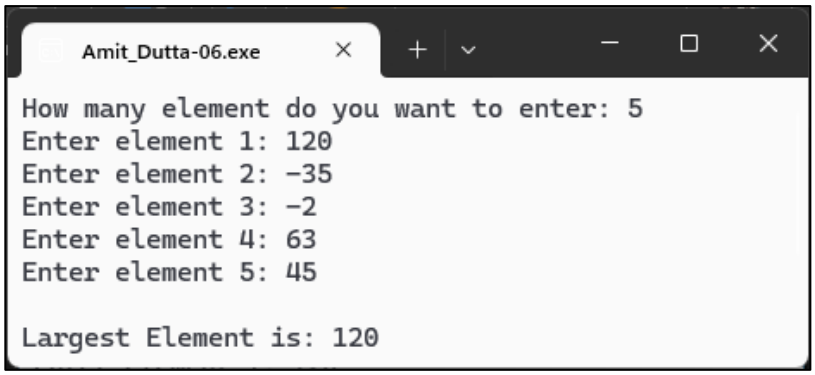
Source Code:

```
#include <stdio.h>

void inputArray(int[], int);
int findLargest(int[], int);

int main()
{
    int size;
    printf("How many element do you want to enter: ");
    scanf("%d", &size);
    int arr[size];
    inputArray(arr, size);
    printf("\nLargest Element is: %d", findLargest(arr, size));
    return 0;
}

void inputArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
    {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &arr[i]);
    }
}

int findLargest(int arr[], int size)
{
    int largest = arr[0], i;
    for (i = 1; i < size; i++)
    {
        if (largest < arr[i])
        {
            largest = arr[i];
        }
    }
    return largest;
}
```

Ouptut:



Output:


Amit_Dutta-06.exe
How many element do you want to enter: 5
Enter element 1: 120
Enter element 2: -35
Enter element 3: -2
Enter element 4: 63
Enter element 5: 45
Largest Element is: 120

Assignment 7:

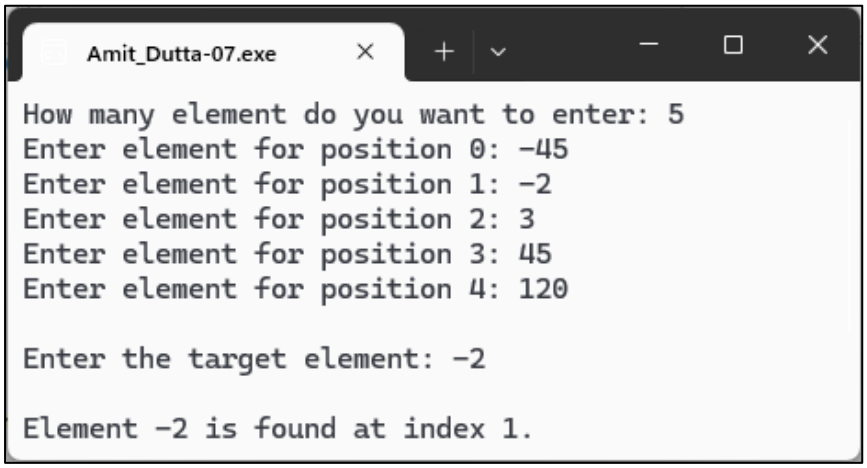
Write a C program that includes a user-defined function named binarySearch with the signature **int binarySearch(int arr[], int size, int target);**. The function should perform a binary search on a sorted array of integers and return the index of the target element if found, and -1 otherwise.

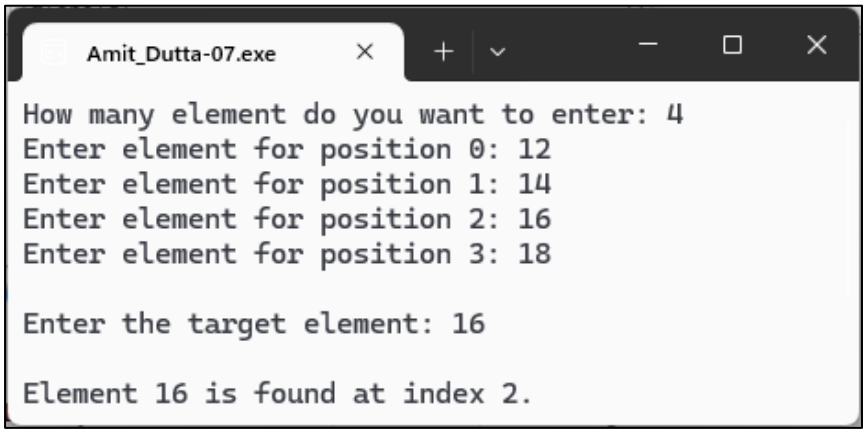
Source Code:

```
#include <stdio.h>

int inputArray(int[], int);
int binarySearch(int[], int, int);

int main()
{
    int size;
    printf("How many element do you want to enter: ");
    scanf("%d", &size);
    int arr[size];
    int target = inputArray(arr, size);
    int index = binarySearch(arr, size, target);
    if (index != -1)
    {
        printf("\nElement %d is found at index %d.", target, index);
    }
    else
    {
        printf("\nElement %d is not found.", target);
    }
    return 0;
}

int inputArray(int arr[], int size)
{
    int i, target;
    for (i = 0; i < size; i++)
    {
        printf("Enter element for position %d: ", i);
        scanf("%d", &arr[i]);
    }
    printf("\nEnter the target element: ");
    scanf("%d", &target);
    return target;
}

int binarySearch(int arr[], int size, int target)
{
    int low = 0;
    int high = size - 1;
    int mid;
    while (low <= high)
    {
        mid = low + ((high - low) / 2);
        if (arr[mid] == target)
        {
            return mid;
```

```
        }
        else if (arr[mid] > target)
        {
            high = mid - 1;
        }
        else if (arr[mid] < target)
        {
            low = mid + 1;
        }
    }
    return -1;
}
```

Output:





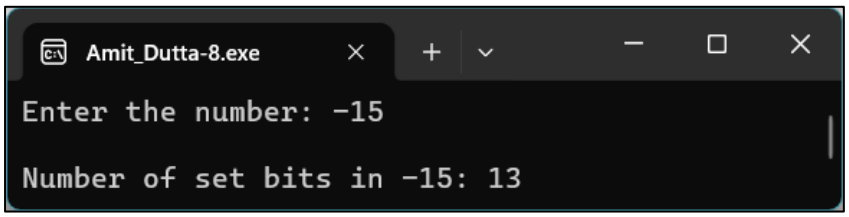
Assignment 8:

Write a C program that includes a user-defined function named countSetBits with the signature **int countSetBits(int num);**. The function should count and return the number of set bits (1s) in the binary representation of the given number.

Source Code:

```
#include <stdio.h>
int countSetBits(int);
int main()
{
    int num, result;
    printf("Enter the number: ");
    scanf("%d", &num);
    if (result = countSetBits(num))
    {
        printf("\nNumber of set bits in %d: %d", num, result);
    }
    else
    {
        printf("\nThere is no set bits in %d", num);
    }
    return 0;
}
int countSetBits(int num)
{
    int count = 0;
    int mask= 1;
    int i = 1;
    while (i <= 16)
    {
        if (num & mask)
        {
            count++;
        }
        mask <<= 1;
        i++;
    }
    return count;
}
```

Output:



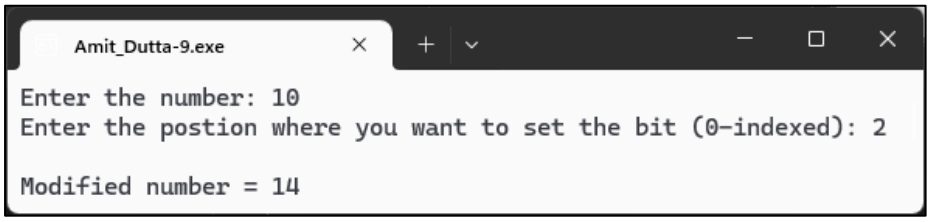
Assignment 9:

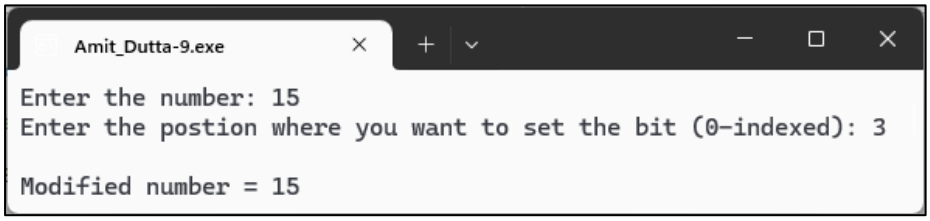
Write a C program that includes a user-defined function named setBit with the signature **int setBit(int num, int position);**. The function should set the bit at the specified position (0-indexed) to 1 and return the modified number.

Source Code:

```
#include <stdio.h>

int setBit(int, int);

int main()
{
    int num, position;
    printf("Enter the number: ");
    scanf("%d", &num);
    printf("Enter the postion where you want to set the bit (0-indexed): ");
    scanf("%d", &position);
    printf("\nModified number = %d", setBit(num, position));
    return 0;
}

int setBit(int num, int position)
{
    int mask = 1 << position;
    return num | mask;
}
```

Output:





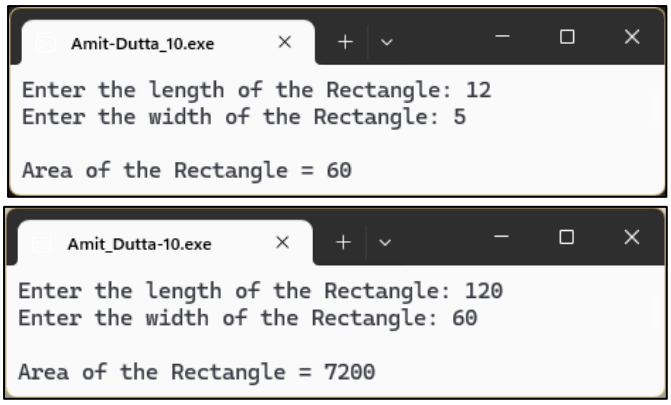
Assignment 10:

Write a C program that defines a structure Rectangle with attributes length and width. Include a user-defined function named calculateArea with the signature **float calculateArea(struct Rectangle r);**. The function should calculate and return the area of the rectangle.

Source Code:

```
#include <stdio.h>

struct Rectangle
{
    float length;
    float width;
};

float calculateArea(struct Rectangle);

int main()
{
    struct Rectangle rectangle;
    printf("Enter the length of the Rectangle: ");
    scanf("%f", &rectangle.length);
    printf("Enter the width of the Rectangle: ");
    scanf("%f", &rectangle.width);
    printf("\nArea of the Rectangle = %g", calculateArea(rectangle));
}

float calculateArea(struct Rectangle r)
{
    return r.length * r.width;
}
```

Output:



Assignment 11:

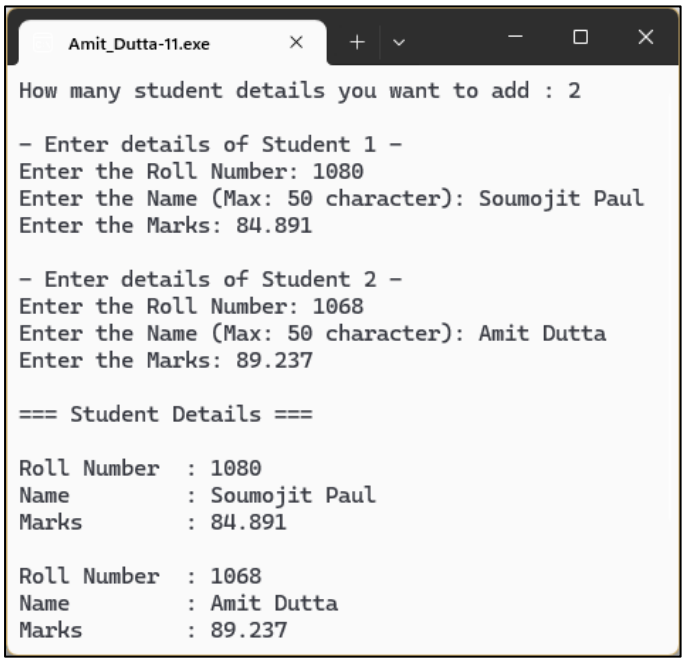
Write a C program that defines a structure Student containing the attributes rollNumber, name, and marks. Include a user-defined function named displayStudent with the signature **void displayStudent(struct Student s);**. The function should display the details of a student.

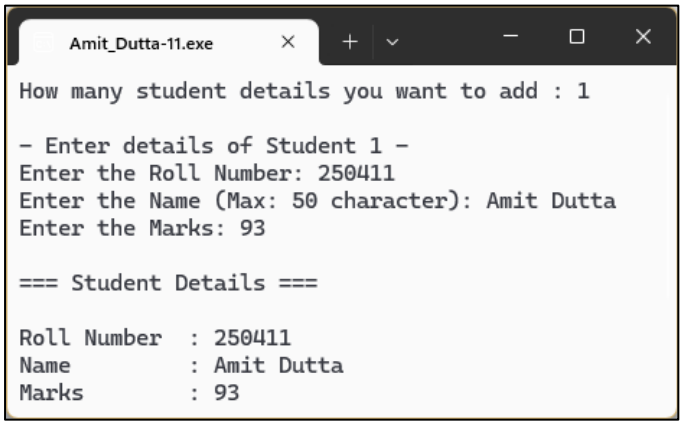
Source Code:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
struct Student
{
    int rollNumber;
    char name[50];
    float marks;
};
void inputStudent(struct Student *);
void displayStudent(struct Student);
int main()
{
    struct Student *std = NULL;
    int i, n;

    printf("How many student details you want to add : ");
    if (scanf("%d", &n) != 1 || n < 1)
    {
        printf("\nInvalid Input.");
        return 1;
    }

    std = (struct Student *)malloc(n * sizeof(struct Student));
    if (std == NULL)
    {
        printf("\nUnable to allocate memory.");
        return 1;
    }

    for (i = 0; i < n; i++)
    {
        printf("\n- Enter details of Student %d -", i + 1);
        inputStudent(&std[i]);
    }

    printf("\n=== Student Details ===\n");
    for (i = 0; i < n; i++)
    {
        displayStudent(std[i]);
    }
    free(std);
    return 0;
}
void inputStudent(struct Student *std)
{
    int len;
```

```
    printf("\nEnter the Roll Number: ");
    scanf("%d", &std->rollNumber);
    getchar();

    printf("Enter the Name (Max: 50 character): ");
    fgets(std->name, sizeof(std->name), stdin);
    len = strlen(std->name);
    if (len > 0 && std->name[len - 1] == '\n')
    {
        std->name[len - 1] = '\0';
    }

    printf("Enter the Marks: ");
    scanf("%f", &std->marks);
}
void displayStudent(struct Student std)
{
    printf("\n%-12s : %d", "Roll Number", std.rollNumber);
    printf("\n%-12s : %s", "Name", std.name);
    printf("\n%-12s : %g\n", "Marks", std.marks);
}
```

Output:





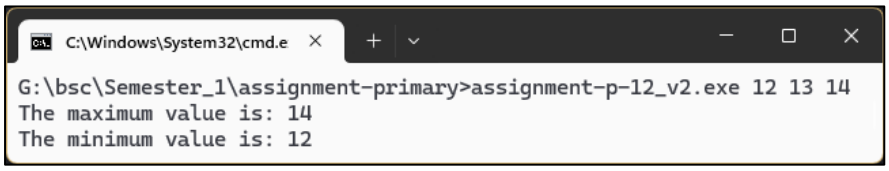
Assignment 12:

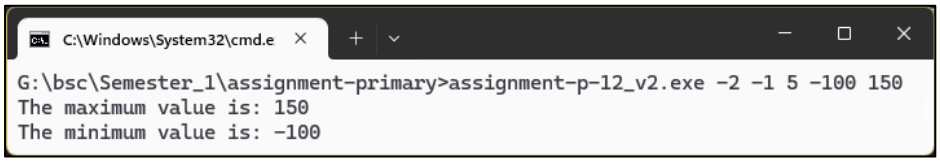
Write a C program that takes multiple integers as command-line arguments and finds the maximum and minimum value among them.

Source Code:

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

int main(int argc, char *argv[])
{
    int current_val, max_val, min_val, i;
    if (argc < 2)
    {
        printf("Usage: %s <integer1> <integer2> ...\n", argv[0]);
        return 1;
    }
    max_val = atoi(argv[1]);
    min_val = atoi(argv[1]);
    for (i = 2; i < argc; i++)
    {
        current_val = atoi(argv[i]);

        if (current_val > max_val)
        {
            max_val = current_val;
        }
        if (current_val < min_val)
        {
            min_val = current_val;
        }
    }
    printf("The maximum value is: %d\n", max_val);
    printf("The minimum value is: %d\n", min_val);
    return 0;
}
```

Output:





Assignment 13:

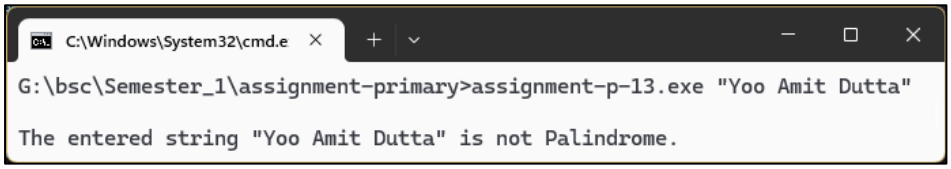
Write a C program that accepts a string as a command line argument and includes a user-defined function named isPalindrome with the signature **int isPalindrome(char str[]);**. The function should check if the given string is a palindrome and return 1 if it is, and 0 otherwise.

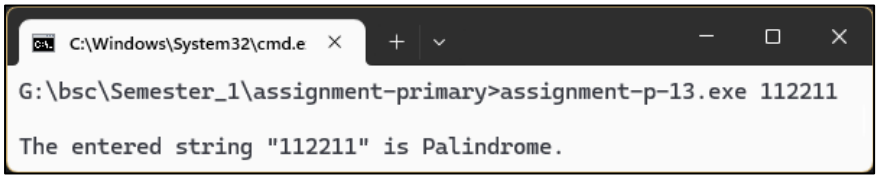
Source code:

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

int isPalindrome(char[]);

int main(int argc, char *argv[])
{
    if (argc != 2)
    {
        printf("\nUsage: %s <string>\n", argv[0]);
        return 1;
    }

    if (isPalindrome(argv[1]))
    {
        printf("\nThe entered string \"%s\" is Palindrome.\n", argv[1]);
    }

    else
    {
        printf("\nThe entered string \"%s\" is not Palindrome.\n", argv[1]);
    }

    return 0;
}

int isPalindrome(char str[])
{
    char *start = str;
    char *end;
    int len = strlen(str);

    if (len == 0)
    {
        return 1;
    }
    end = str + (len - 1);

    while (start < end)
    {
        if (*start != *end)
        {
            return 0;
        }
        start++;
```

```
        end--;
    }
    return 1;
}
```

Output:





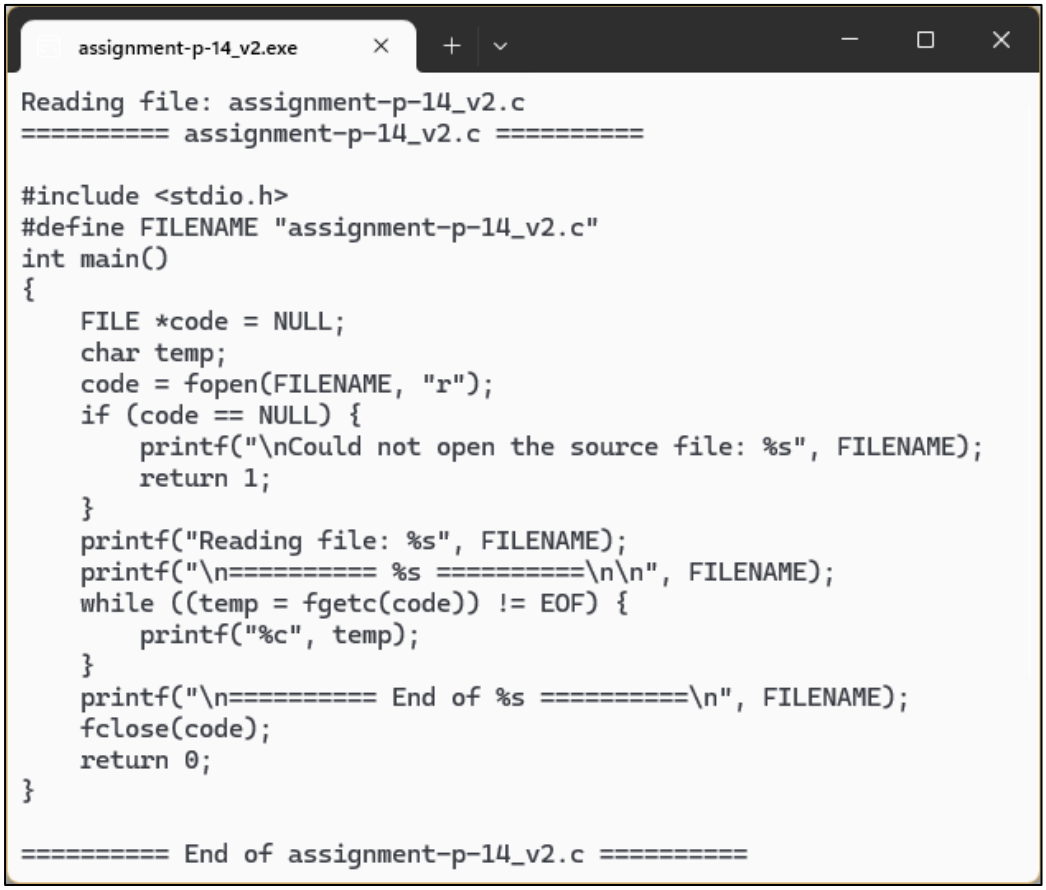
Assignment 14:

Write a C program that opens its own source code file, reads its contents, and then prints the contents to the console.

Source Code:

```
#include <stdio.h>
#define FILENAME "assignment-p-14_v2.c"
int main()
{
    FILE *code = NULL;
    char temp;
    code = fopen(FILENAME, "r");
    if (code == NULL) {
        printf("\nCould not open the source file: %s", FILENAME);
        return 1;
    }
    printf("Reading file: %s", FILENAME);
    printf("\n========== %s ==========\n\n", FILENAME);
    while ((temp = fgetc(code)) != EOF) {
        printf("%c", temp);
    }
    printf("\n========== End of %s ==========\n", FILENAME);
    fclose(code);
    return 0;
}
```

Output:



Assignment 15:

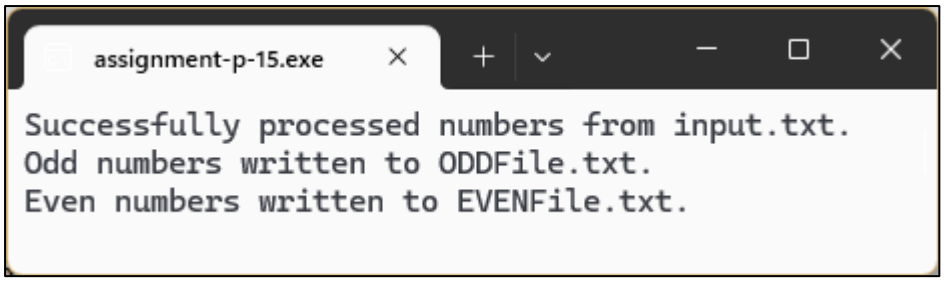
Write a C program that reads a sequence of integers from a file named '**input.txt**'. This program should segregate the odd numbers from the even numbers and store the odd numbers in a new file named '**ODDFile.txt**' while storing the even numbers in another file named '**EVENFile.txt**'.

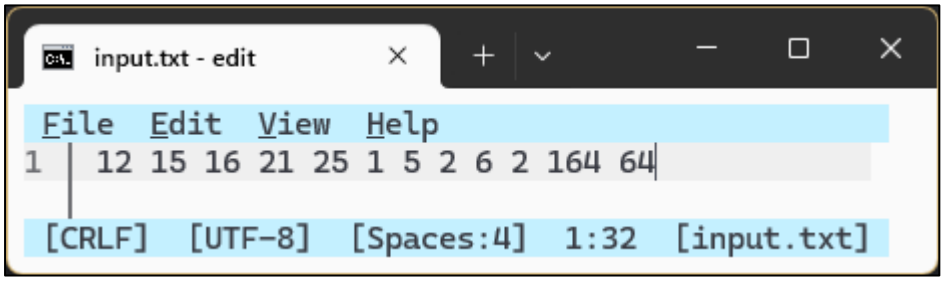
Source Code:

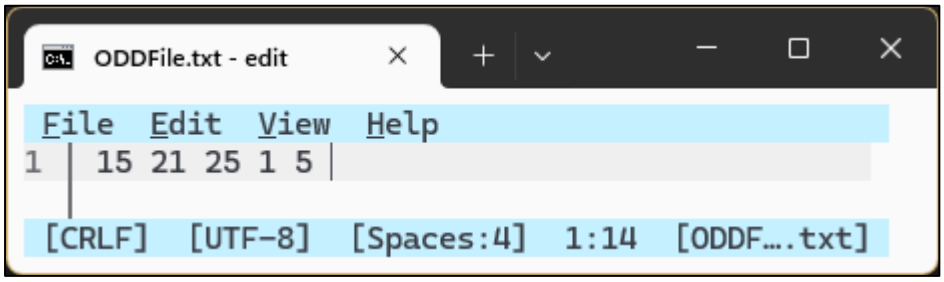
```
#include <stdio.h>

#define FILENAME "input.txt"
#define ODDFILE "ODDFile.txt"
#define EVENFILE "EVENFile.txt"

int main()
{
    FILE *input = NULL;
    FILE *oddfile = NULL;
    FILE *evenfile = NULL;

    int num;

    input = fopen(FILENAME, "r");
    if (input == NULL)
    {
        printf("\nCould not open the file: %s", FILENAME);
        return 1;
    }

    oddfile = fopen(ODDFILE, "w");
    if (oddfile == NULL)
    {
        printf("\nCould not open the file: %s", ODDFILE);
        return 1;
    }

    evenfile = fopen(EVENFILE, "w");
    if (evenfile == NULL)
    {
        printf("\nCould not open the file: %s", EVENFILE);
        return 1;
    }

    while (fscanf(input, "%d", &num) == 1)
    {
        if (num % 2 == 0)
        {
            fprintf(evenfile, "%d ", num);
        }
        else
        {
            fprintf(oddfile, "%d ", num);
        }
    }
```

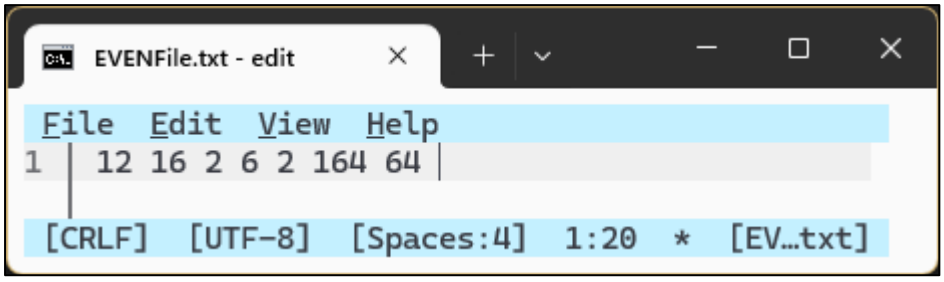
```
    printf("Successfully processed numbers from %s.\n", FILENAME);
    printf("Odd numbers written to %s.\n", ODDFILE);
    printf("Even numbers written to %s.\n", EVENFILE);

    fclose(input);
    fclose(oddfile);
    fclose(evenfile);

    return 0;
}
```

Output:









Assignment 15:

Write a C program that reads a sequence of integers from a file named '**input.txt**'. This program should segregate the odd numbers from the even numbers and store the odd numbers in a new file named '**ODDFile.txt**' while storing the even numbers in another file named '**EVENFile.txt**'.

Source Code:

```
#include <stdio.h>

#define FILENAME "input.txt"
#define ODDFILE "ODDFile.txt"
#define EVENFILE "EVENFile.txt"

int main()
{
    FILE *input = NULL;
    FILE *oddfile = NULL;
    FILE *evenfile = NULL;

    int num;

    input = fopen(FILENAME, "r");
    if (input == NULL)
    {
        printf("\nCould not open the file: %s", FILENAME);
        return 1;
    }

    oddfile = fopen(ODDFILE, "w");
    if (oddfile == NULL)
    {
        printf("\nCould not open the file: %s", ODDFILE);
        return 1;
    }

    evenfile = fopen(EVENFILE, "w");
    if (evenfile == NULL)
    {
        printf("\nCould not open the file: %s", EVENFILE);
        return 1;
    }

    while (fscanf(input, "%d", &num) == 1)
    {
        if (num % 2 == 0)
        {
            fprintf(evenfile, "%d ", num);
        }
        else
        {
            fprintf(oddfile, "%d ", num);
        }
    }
```

```
    printf("Successfully processed numbers from %s.\n", FILENAME);
    printf("Odd numbers written to %s.\n", ODDFILE);
    printf("Even numbers written to %s.\n", EVENFILE);

    fclose(input);
    fclose(oddfile);
    fclose(evenfile);

    return 0;
}
```

Output: