

### **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:**

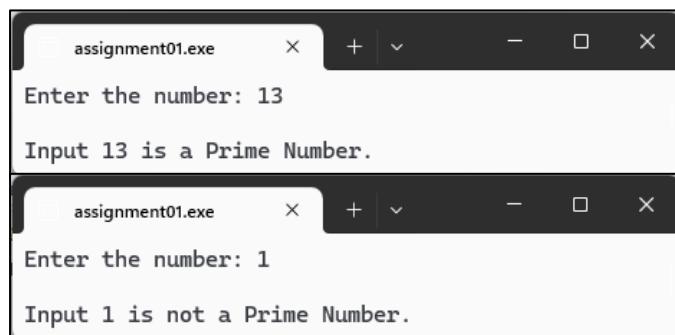
```
#include <stdio.h>
#include <math.h>

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:**



```
assignment01.exe      + | - | X
Enter the number: 13
Input 13 is a Prime Number.

assignment01.exe      + | - | X
Enter the number: 1
Input 1 is not a Prime Number.
```

## **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;
}

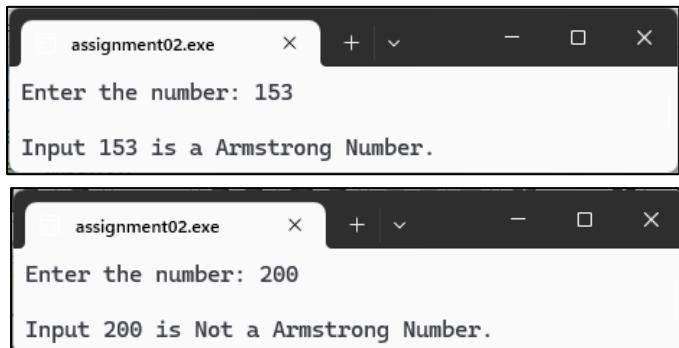
int isArmstrong(int n)
{
    if (n < 0)
        return 0;
    if (n == 0)
        return 1;

    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:**



The image shows two screenshots of a terminal window titled "assignment02.exe". The first screenshot shows the output for the number 153: "Enter the number: 153" followed by "Input 153 is a Armstrong Number.". The second screenshot shows the output for the number 200: "Enter the number: 200" followed by "Input 200 is Not a Armstrong Number.".

```
assignment02.exe  
Enter the number: 153  
Input 153 is a Armstrong Number.  
  
assignment02.exe  
Enter the number: 200  
Input 200 is Not a Armstrong Number.
```

### **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:**

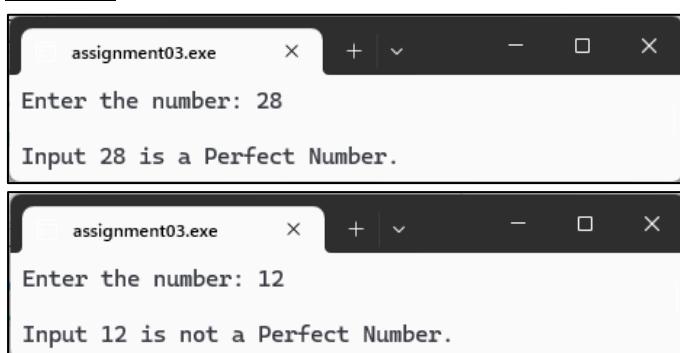
```
#include <stdio.h>

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:**



The image contains two screenshots of a terminal window titled "assignment03.exe".  
Screenshot 1:  
The terminal displays the following text:  
Enter the number: 28  
Input 28 is a Perfect Number.  
Screenshot 2:  
The terminal displays the following text:  
Enter the number: 12  
Input 12 is not a Perfect Number.

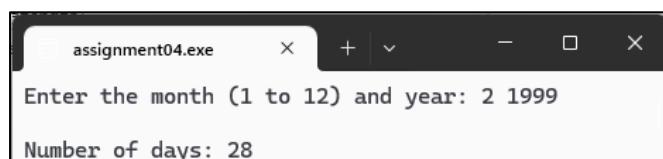
#### **Assignment 4:**

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;
}
```

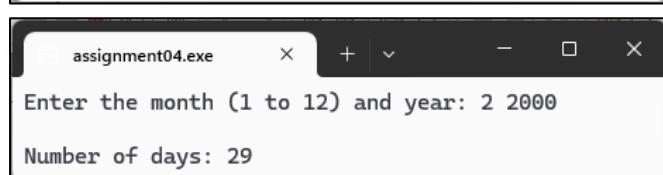
#### **Output:**



assignment04.exe

Enter the month (1 to 12) and year: 2 1999

Number of days: 28



assignment04.exe

Enter the month (1 to 12) and year: 2 2000

Number of days: 29

### **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  X + | - □ X

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  X + | - □ X

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>

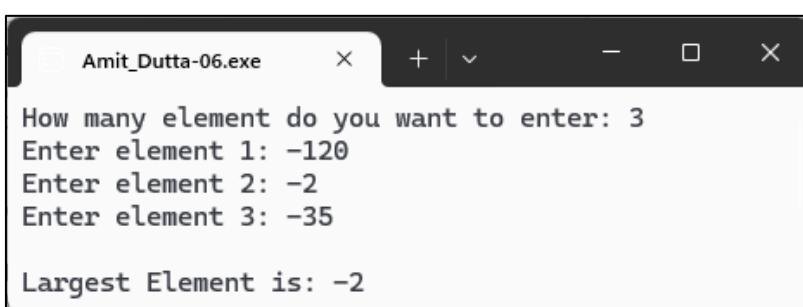
void inputArray(int[], int);
int findLargest(int[], int);

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:**

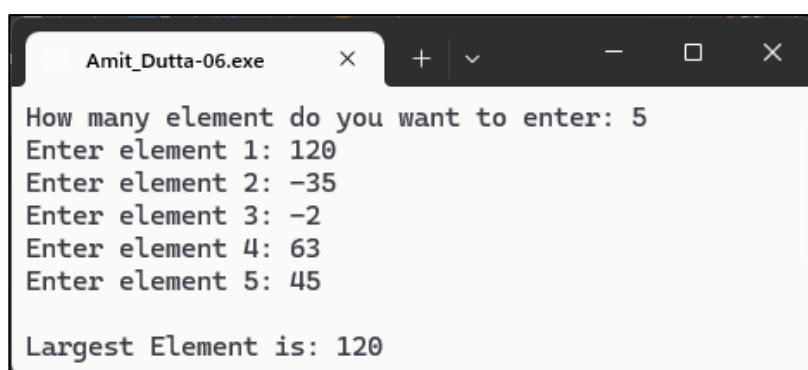


The screenshot shows a terminal window with the title "Amit\_Dutta-06.exe". The window contains the following text:

```
How many element do you want to enter: 3
Enter element 1: -120
Enter element 2: -2
Enter element 3: -35

Largest Element is: -2
```

**Output:**



```
Amit_Dutta-06.exe      X + | - □ ×

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);

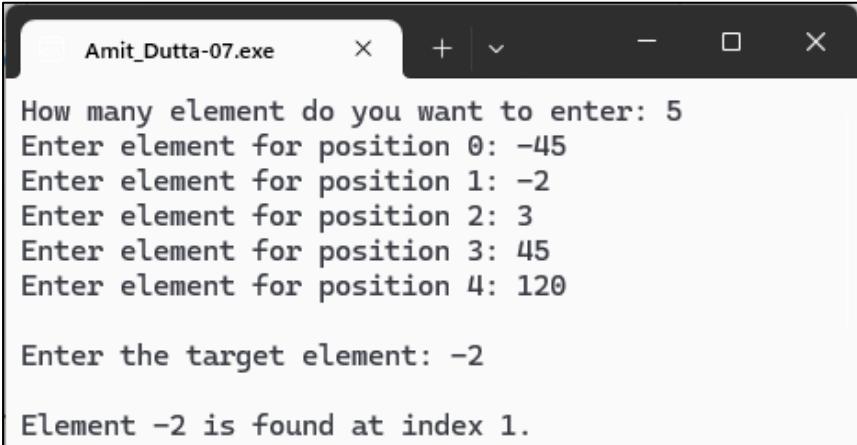
int main()
{
    int size;
    printf("How many element do you want to enter: ");
    scanf("%d", &size);
    int arr[size];
    int target = inputArray(arr, size);
    int index = binarySearch(arr, size, target);
    if (index != -1)
    {
        printf("\nElement %d is found at index %d.", target, 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;
}
```

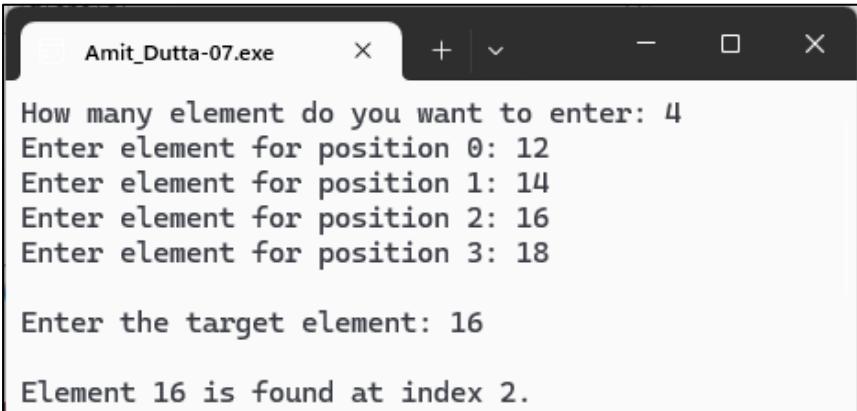
**Output:**



```
Amit_Dutta-07.exe
How many element do you want to enter: 5
Enter element for position 0: -45
Enter element for position 1: -2
Enter element for position 2: 3
Enter element for position 3: 45
Enter element for position 4: 120

Enter the target element: -2

Element -2 is found at index 1.
```



```
Amit_Dutta-07.exe
How many element do you want to enter: 4
Enter element for position 0: 12
Enter element for position 1: 14
Enter element for position 2: 16
Enter element for position 3: 18

Enter the target element: 16

Element 16 is found at index 2.
```

### **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.

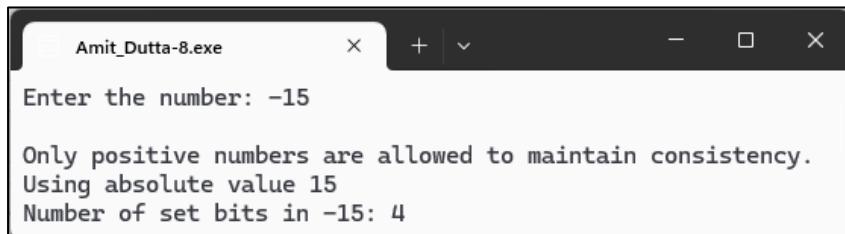
### **Source Code:**

```
#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;
    if (num < 0)
    {
        printf("\nOnly positive numbers are allowed to maintain consistency. \nUsing
absolute value %d", -num);
        num = -num;
    }
    while (num > 0)
    {
        if (num & 1)
        {
            count++;
        }
        num >>= 1;
    }
    return count;
}
```

### **Output:**



```
Amit_Dutta-8.exe
Enter the number: 12
Number of set bits in 12: 2
```



```
Amit_Dutta-8.exe
Enter the number: -15
Only positive numbers are allowed to maintain consistency.
Using absolute value 15
Number of set bits in -15: 4
```

### **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:**

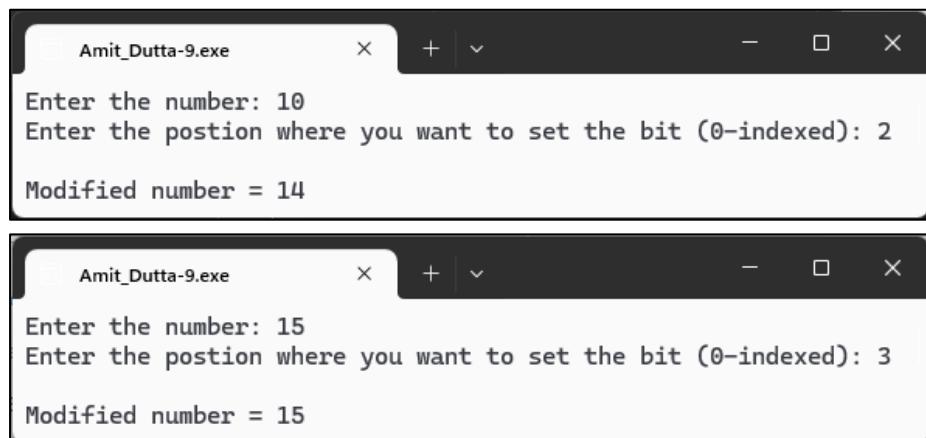
```
#include <stdio.h>

int setBit(int, int);

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



```
Amit_Dutta-9.exe      + | - X
Enter the number: 10
Enter the position where you want to set the bit (0-indexed): 2
Modified number = 14

Amit_Dutta-9.exe      + | - X
Enter the number: 15
Enter the position where you want to set the bit (0-indexed): 3
Modified number = 15
```