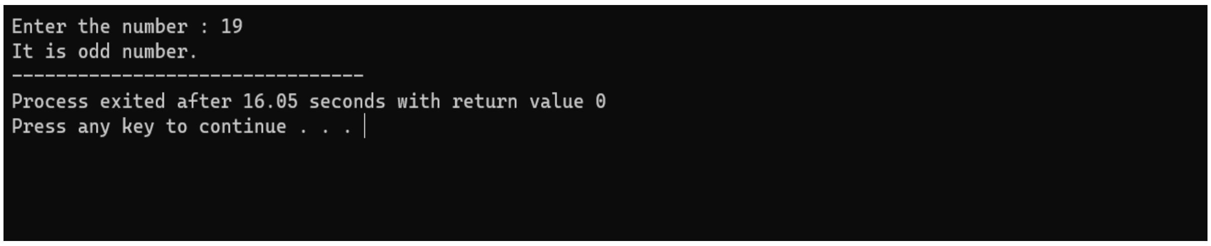
DAY-1

1.Write a c program to check given number is odd or not?

Program :

```
#include<stdio.h>

int main ()

{

        int n;

        {

        Printf ("Enter the number : ");

        }

        Scanf ("%d",&n);

        if (n%2==0)

        printf ("It is even number.");

        else

        printf ("It is odd number.");

        return 0;

}
```

Output:



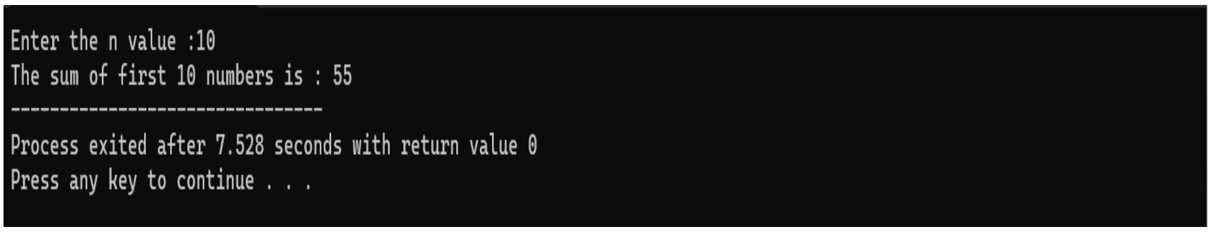
```
Enter the number : 19
It is odd number.
--------------------------------
Process exited after 16.05 seconds with return value 0
Press any key to continue . . .
```

2.write a c program to find sum of first n numbers using any loop?

Program :

```
#include <stdio.h>

int main ()

{

        int i, range, sum =0;
        printf("Enter the n value :");

        scanf("%d",&range);

        for(i=1; i<=range ; i++){

                sum += i;

        }

        printf("The sum of first %d numbers is : %d", range, sum);

        return 0;

}
```

Output:



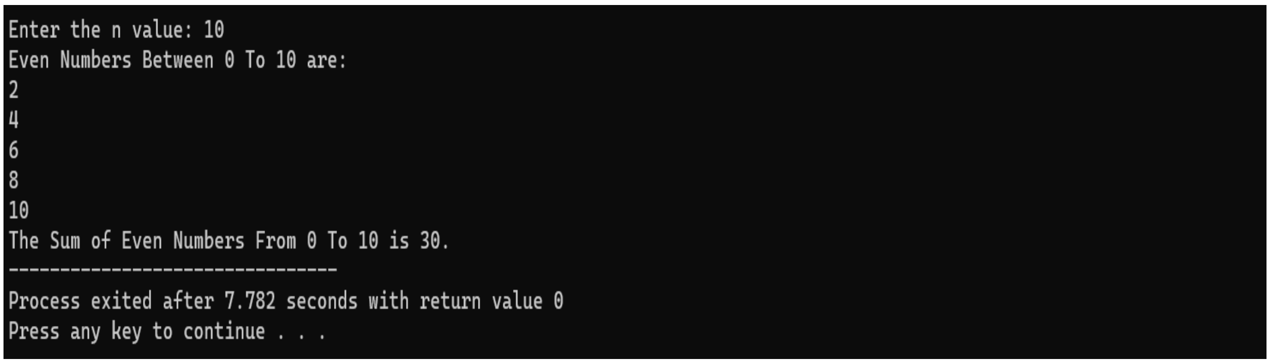
```
Enter the n value :10
The sum of first 10 numbers is : 55
--------------------------------
Process exited after 7.528 seconds with return value 0
Press any key to continue . . .
```

3.write a c program to find sum of even numbers in the first n numbers?

Program :

```
#include <stdio.h>

int main(){

  int i, num, sum = 0;

  printf("Enter the n value: ");

  scanf("%d", &num);

  printf("Even Numbers Between 0 To %d are: \n", num);

  for (i = 1; i <= num; i++ ){

    if (i % 2 == 0){

      printf("%d\n", i);

      sum = sum + i;

    }

  }

  printf("The Sum of Even Numbers From 0 To %d is %d.", num, sum);

  return 0;

}
```

Output :



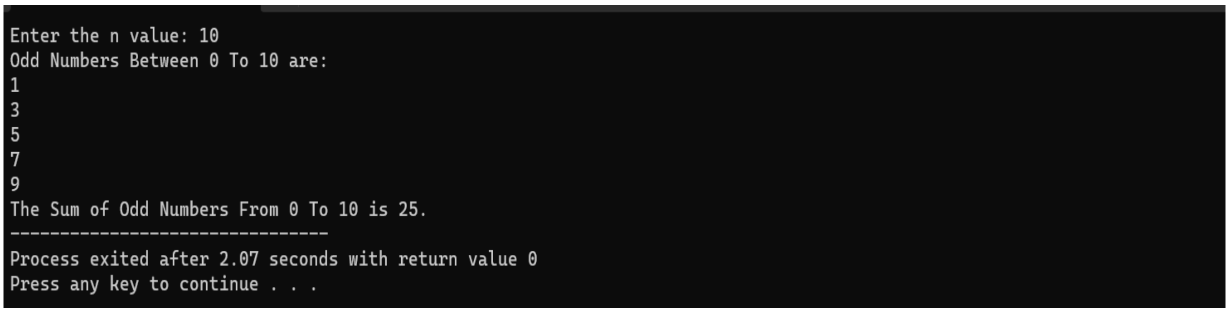
```
Enter the n value: 10
Even Numbers Between 0 To 10 are:
2
4
6
8
10
The Sum of Even Numbers From 0 To 10 is 30.
--------------------------------
Process exited after 7.782 seconds with return value 0
Press any key to continue . . .
```

4. write a c program to find sum of odd numbers in the first n numbers?

Program :

```
#include <stdio.h>

int main(){

   int i, num, sum = 0;

   printf("Enter the n value: ");

   scanf("%d", &num);

   printf("Odd Numbers Between 0 To %d are: \n", num);

   for (i = 1; i <= num; i++ ){

      if (i % 2 != 0){

         printf("%d\n", i);

         sum = sum + i;

      }

   }

   printf("The Sum of Odd Numbers From 0 To %d is %d.", num, sum);

   return 0;

}
```

Output :



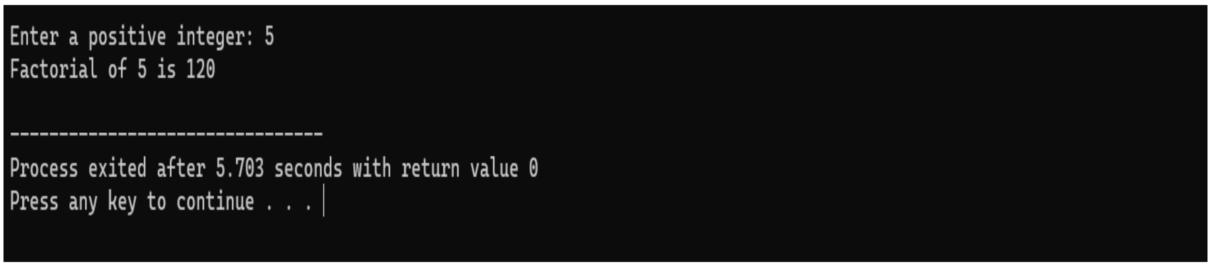
```
Enter the n value: 10
Odd Numbers Between 0 To 10 are:
1
3
5
7
9
The Sum of Odd Numbers From 0 To 10 is 25.
--------------------------------
Process exited after 2.07 seconds with return value 0
Press any key to continue . . .
```

5. write a c program to find factorial of a given number with recursion?

Program :

```
#include <stdio.h>

int factorial(int n) {

   if (n == 0 || n == 1) {

      return 1;

   } else {

      return n * factorial(n - 1);

   }

}

int main() {

   int num;

   printf("Enter a positive integer: ");

   scanf("%d", &num);

   if (num < 0) {

      printf("Factorial is not defined for negative numbers.\n");

   } else {

      int result = factorial(num);

      printf("Factorial of %d is %d\n", num, result);

   }

   return 0;

}
```

Output :



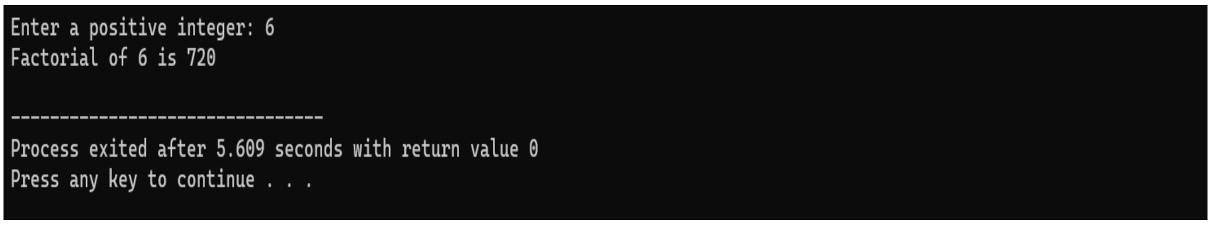
```
Enter a positive integer: 5
Factorial of 5 is 120

--------------------------------
Process exited after 5.703 seconds with return value 0
Press any key to continue . . .
```

6. write a c program to find factorial of a given number without recursion?

Program :

```
#include <stdio.h>

int main() {

   int num ,i;

   unsigned long long factorial = 1;

   printf("Enter a positive integer: ");

   scanf("%d", &num);

   if (num < 0) {

      printf("Factorial is not defined for negative numbers.\n");

   } else {

      for ( i = 1; i <= num; i++) {

         factorial *= i;

      }

      printf("Factorial of %d is %llu\n", num, factorial);

   }

   return 0;

}
```

Output :



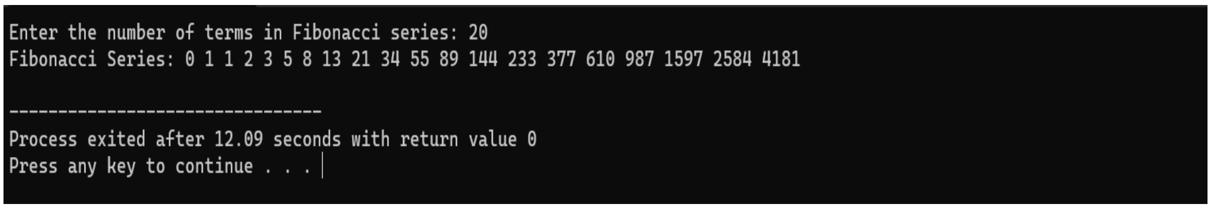
```
Enter a positive integer: 6
Factorial of 6 is 720

--------------------------------
Process exited after 5.609 seconds with return value 0
Press any key to continue . . .
```

7. Write a c program to generate Fibonacci series with recursion?

Program :

```
#include <stdio.h>

int fibonacci(int n) {

    if (n <= 0)

        return 0;

    else if (n == 1)

        return 1;

    else

        return fibonacci(n - 1) + fibonacci(n - 2);

}

int main() {

    int num ,i;

    printf("Enter the number of terms in Fibonacci series: ");

    scanf("%d", &num);

    if (num <= 0) {

        printf("Number of terms should be positive.\n");

    } else {

        printf("Fibonacci Series: ");

        for ( i = 0; i < num; i++) {

            printf("%d ", fibonacci(i));

        }

        printf("\n");

    }

    return 0;

}
```

Output :



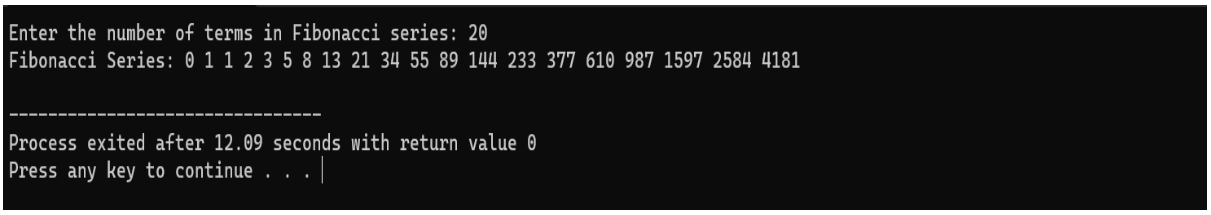
```
Enter the number of terms in Fibonacci series: 20
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181

--------------------------------
Process exited after 12.09 seconds with return value 0
Press any key to continue . . .
```

8. Write a c program to generate Fibonacci series without recursion?

Program :

```
#include <stdio.h>

int main() {

    int num ,i;

    printf("Enter the number of terms in Fibonacci series: ");

    scanf("%d", &num);

    if (num <= 0) {

        printf("Number of terms should be positive.\n");

    } else {

        int fib[num];

        fib[0] = 0;

        fib[1] = 1;

        printf("Fibonacci Series: %d %d ", fib[0], fib[1]);

        for ( i = 2; i < num; i++) {

            fib[i] = fib[i - 1] + fib[i - 2];

            printf("%d ", fib[i]);

        }

        printf("\n");

    }

    return 0;

}
```

Output :



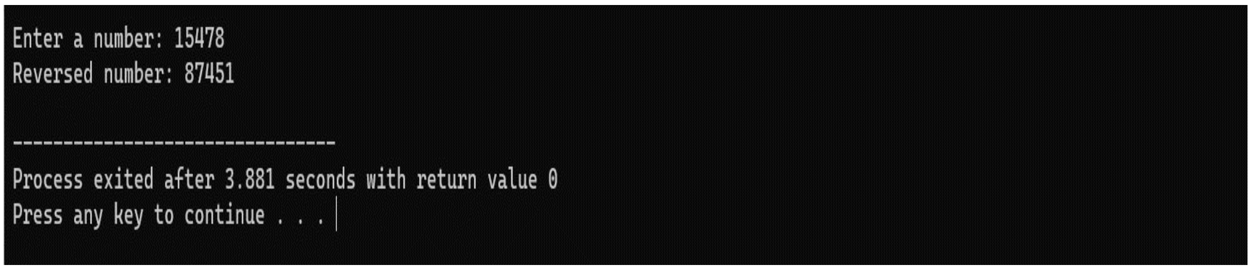
```
Enter the number of terms in Fibonacci series: 20
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181

--------------------------------
Process exited after 12.09 seconds with return value 0
Press any key to continue . . .
```

9. Write a c program to reverse a number?

Program :

```
#include <stdio.h>

int reverseNumber(int num) {
    int reversedNum = 0;

    while (num > 0) {

        int remainder = num % 10;

        reversedNum = reversedNum * 10 + remainder;

        num /= 10;

    }

    return reversedNum;

}

int main() {

    int num;

    printf("Enter a number: ");

    scanf("%d", &num);

    int reversed = reverseNumber(num);

    printf("Reversed number: %d\n", reversed);

    return 0;

}
```

Output :



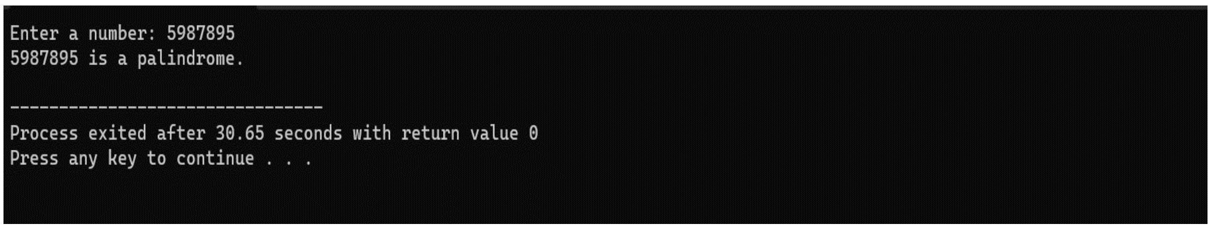
```
Enter a number: 15478
Reversed number: 87451

--------------------------------
Process exited after 3.881 seconds with return value 0
Press any key to continue . . .
```

10. Write a **c program to check the given number is palindrome or not?**

Program :

```
#include <stdio.h>

int isPalindrome(int num) {

    int originalNum = num;

    int reversedNum = 0;

    while (num > 0) {

        int remainder = num % 10;
        reversedNum = reversedNum * 10 + remainder;
        num /= 10;
    }
    if (originalNum == reversedNum) {
        return 1;
    } else {
        return 0;
    }
}
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    if (isPalindrome(num)) {
        printf("%d is a palindrome.\n", num);
    } else {
        printf("%d is not a palindrome.\n", num);
    }
    return 0;
}
```

Output :



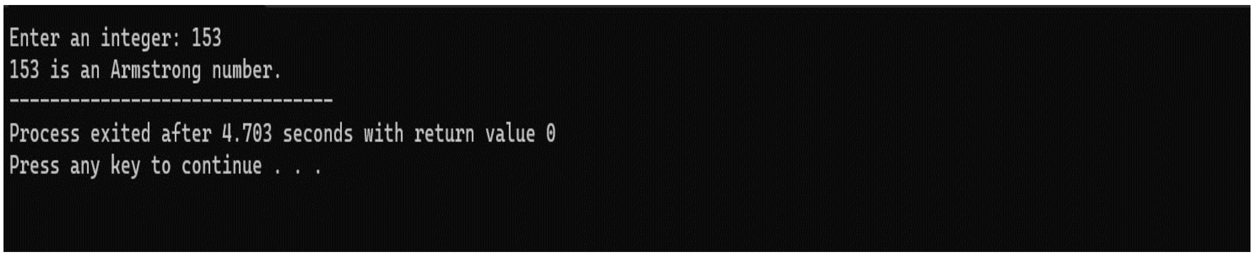
```
Enter a number: 5987895
5987895 is a palindrome.

--------------------------------
Process exited after 30.65 seconds with return value 0
Press any key to continue . . .
```

11. Write a c program to check the given number is Armstrong or not?

Program:

```
#include <stdio.h>
#include <math.h>
int main() {
    int n, originalNumber, remainder, result = 0, nDigits = 0;

    printf("Enter an integer: ");

    scanf("%d", &n);

    originalNumber = n;

    while (originalNumber != 0) {

        originalNumber /= 10;

        ++nDigits;

    }

    originalNumber = n;

    while (originalNumber != 0) {

        remainder = originalNumber % 10;

        result += pow(remainder, nDigits);

        originalNumber /= 10;

    }

    if (result == n) {

        printf("%d is an Armstrong number.", n);

    } else {

        printf("%d is not an Armstrong number.", n);

    }

    return 0;

}
```

Output :



```
Enter an integer: 153
153 is an Armstrong number.
--------------------------------
Process exited after 4.703 seconds with return value 0
Press any key to continue . . .
```

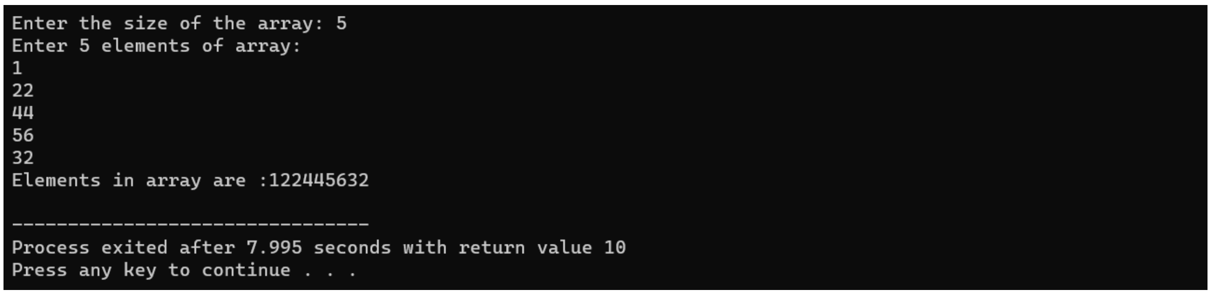
DAY-2

1.write a c program to initialize array and print the array?

Program:

```
#include <stdio.h>

int main (){

        int size ,i;

        printf("Enter the size of the array: ");

        scanf("%d",&size);

        int array[size];

        printf("Enter %d elements of array: \n", size);

        for (i = 0; i < size; i++){

                scanf("%d", &array[i]);

        }

        printf("Elements in array are :");

        for (i = 0;i < size; i++){

                printf("%d", array[i]);

        }

        printf("\n");

}
```

Output:



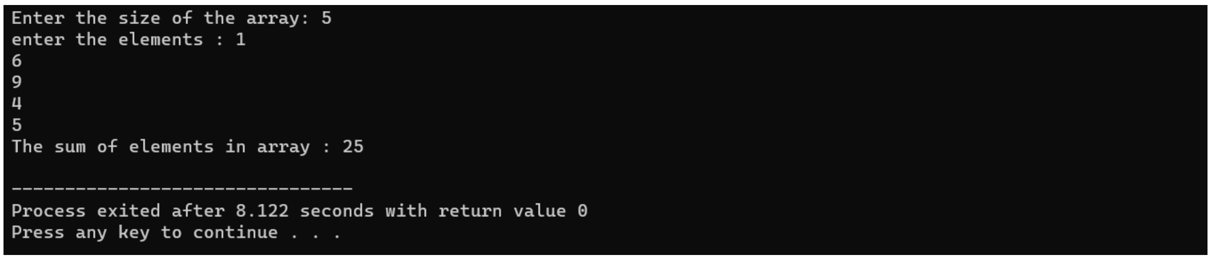
```
Enter the size of the array: 5
Enter 5 elements of array:
1
22
44
56
32
Elements in array are :122445632

--------------------------------
Process exited after 7.995 seconds with return value 10
Press any key to continue . . .
```

2.write a c program to find sum of elements in the given array?

Program:

```
#include <stdio.h>

int main ()

{

        int size ,i;

        printf("Enter the size of the array: ");
        scanf("%d",&size);

        int array[size];

        printf("enter the elements : ");

        for(i = 0; i < size; i++){

                scanf("%d", &array[i]);

        }

        int sum=0;

        for(i =0;i < size;i++){

                sum += array[i];

        }

        printf("The sum of elements in array : %d\n",sum);

        return 0;

}
```

Output:



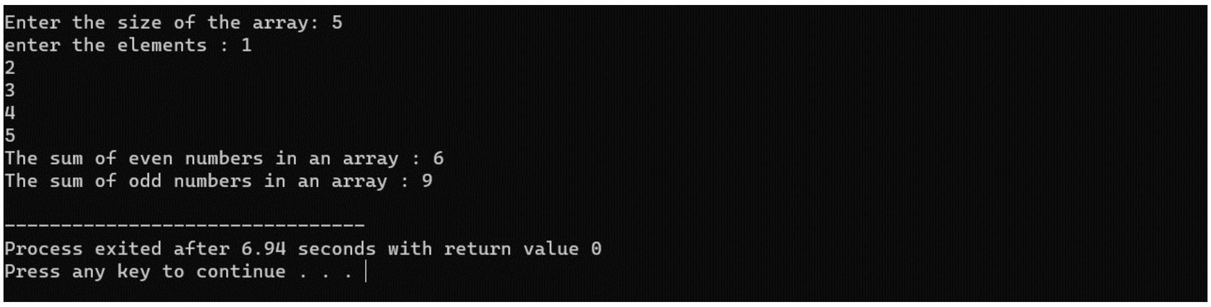
```
Enter the size of the array: 5
enter the elements : 1
6
9
4
5
The sum of elements in array : 25

--------------------------------
Process exited after 8.122 seconds with return value 0
Press any key to continue . . .
```

3.write a c program to find sum of even and sum of odd numbers in an array?

Program:

```
#include <stdio.h>

int main ()

{

        int size ,i;

        printf("Enter the size of the array: ");

        scanf("%d",&size);

        int array[size];

        printf("enter the elements : ");

        for(i = 0; i < size; i++){
                scanf("%d", &array[i]);
        }
        int sumEven=0;
        int sumOdd=0;
        for(i =0;i < size;i++){
                if(array[i] %2==0){
                        sumEven += array[i];
                }
                else{
                        sumOdd += array [i];
                }
        }
        printf("The sum of even numbers in an array : %d\n",sumEven);
        printf("The sum of odd numbers in an array : %d\n",sumOdd);
        return 0;
}
```

Output:



```
Enter the size of the array: 5
enter the elements : 1
2
3
4
5
The sum of even numbers in an array : 6
The sum of odd numbers in an array : 9

--------------------------------
Process exited after 6.94 seconds with return value 0
Press any key to continue . . .
```

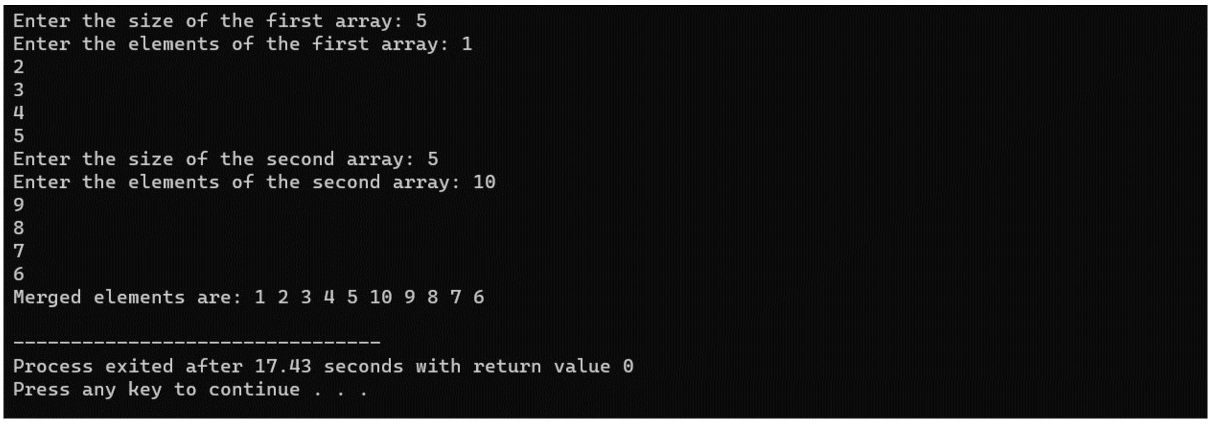
4.write a c program to merge the two array of elements?

Program:

```
#include <stdio.h>
int main ()
{
        int size1,size2,i;
                printf("Enter the size of first array: ");
        scanf("%d",&size1);
```

```
	int array1[size1];
	printf("Enter the elements of first array : ");
	for(i = 0;i < size1;i++){
		scanf("%d",array1[i]);
	}
	printf("Enter the size of second array: ");
	scanf("%d",&size2);
	int array2[size2];
	printf("Enter the elements of second array : ");
	for(i = 0;i < size2;i++){
		scanf("%d",array2[i]);
	}
	int mergedsize = size1+size2;
	int mergedarray[mergedsize];
	for(i = 0;i < size1; i++){
		mergedarray[i]=array1[i];
	}
	for(i = 0;i < size2; i++){
		mergedarray[size1 + i]=array2[i];
	}
	printf("merged elements are : ");
	for(i = 0;i < mergedsize;i++){
		printf("%d",mergedarray[i]);
	}
	printf("\n");
	return 0;
}
```

Output:



```
Enter the size of the first array: 5
Enter the elements of the first array: 1
2
3
4
5
Enter the size of the second array: 5
Enter the elements of the second array: 10
9
8
7
6
Merged elements are: 1 2 3 4 5 10 9 8 7 6

--------------------------------
Process exited after 17.43 seconds with return value 0
Press any key to continue . . .
```

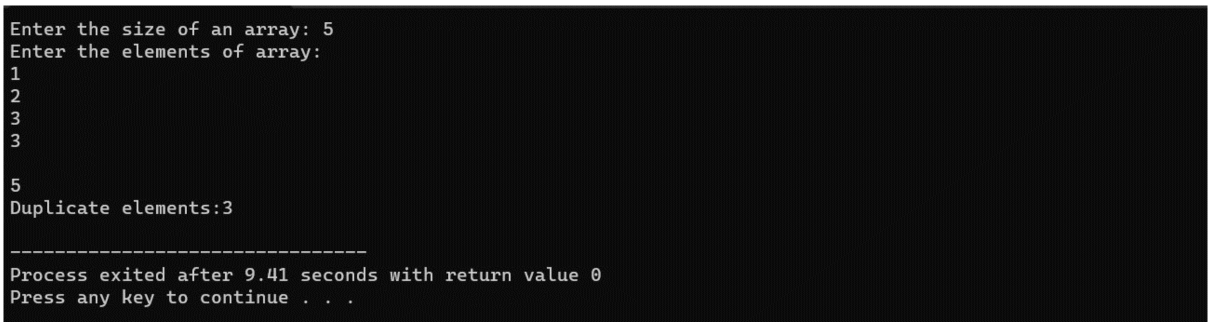
5.Write a c program to find duplicate element in an array?

Program:

```
#include <stdio.h>

int main ()

{

        int size ,i ,j;

        printf("Enter the size of an array: ");

        scanf("%d",&size);

        int array[size];

        printf("Enter the elements of array: \n",size);

        for(i = 0;i < size;i++){

                scanf("%d",&array[i]);

        }

        printf("Duplicate elements:");

        for(i = 0;i < size; i++){

                for(j = i + 1;j < size;j++){

                        if(array[i] == array[j]){

                                printf("%d",array[i]);

                                break;

                        }

                }

        }

        printf("\n");
```

```
    return 0;
}
```

Output:



```
Enter the size of an array: 5
Enter the elements of array:
1
2
3
3

5
Duplicate elements:3

--------------------------------
Process exited after 9.41 seconds with return value 0
Press any key to continue . . .
```

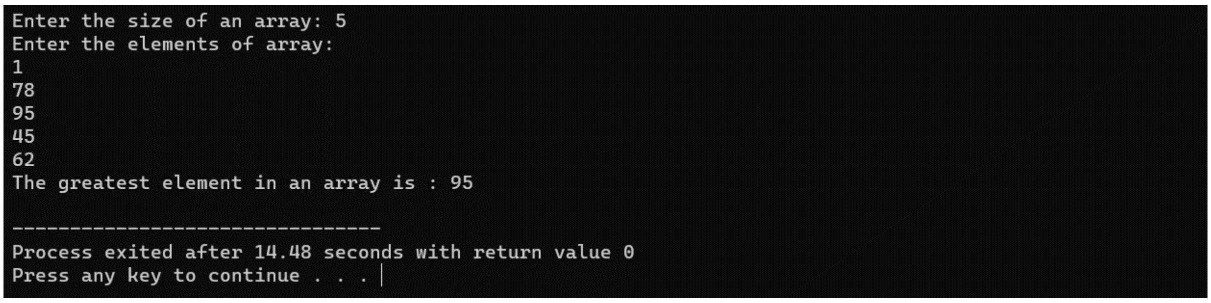
6.Write a c program to find greatest element in an array?

Program:

```
#include <stdio.h>
int main ()
{
    int size ,i ,j;
    printf("Enter the size of an array: ");
    scanf("%d",&size);
    int array[size];
    printf("Enter the elements of array: \n",size);
    for(i = 0;i < size;i++){
        scanf("%d",&array[i]);
    }
    int max = array[0];
    for(i = 0;i < size;i++){
        if(array[i] > max){
            max = array[i];
        }
    }
        printf("The greatest element in an array is : %d\n",max);
    return 0;
```

}

Output:



```
Enter the size of an array: 5
Enter the elements of array:
1
78
95
45
62
The greatest element in an array is : 95

--------------------------------
Process exited after 14.48 seconds with return value 0
Press any key to continue . . .
```

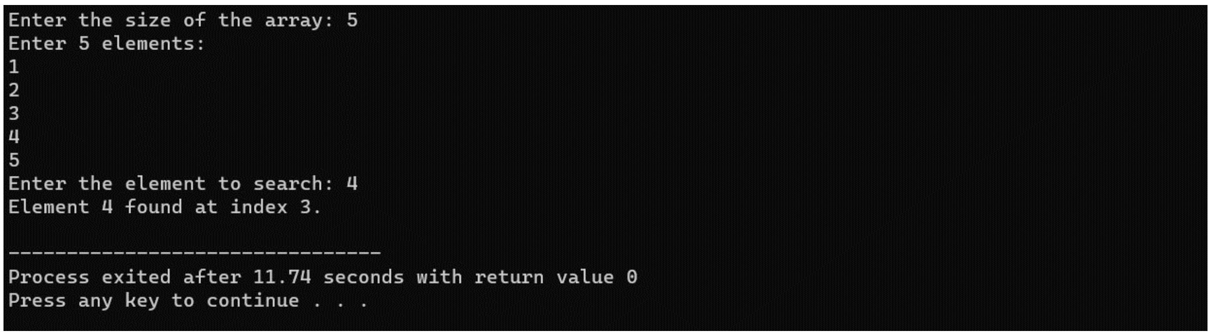
7.Write a c program to find element in an array using linear search?

Program:

```
#include <stdio.h>

int main() {

   int size, key ,i;

   printf("Enter the size of the array: ");

   scanf("%d", &size);

   int array[size];

   printf("Enter %d elements:\n", size);

   for ( i = 0; i < size; i++) {

      scanf("%d", &array[i]);

   }

   printf("Enter the element to search: ");

   scanf("%d", &key);

   int found = 0;

   int index = -1;

   for ( i = 0; i < size; i++) {

      if (array[i] == key) {

         found = 1;

         index = i;

         break;

      }

   }
```

```
    if (found) {
        printf("Element %d found at index %d.\n", key, index);
    } else {
        printf("Element %d not found in the array.\n", key);
    }
    return 0;
}
```

Output:



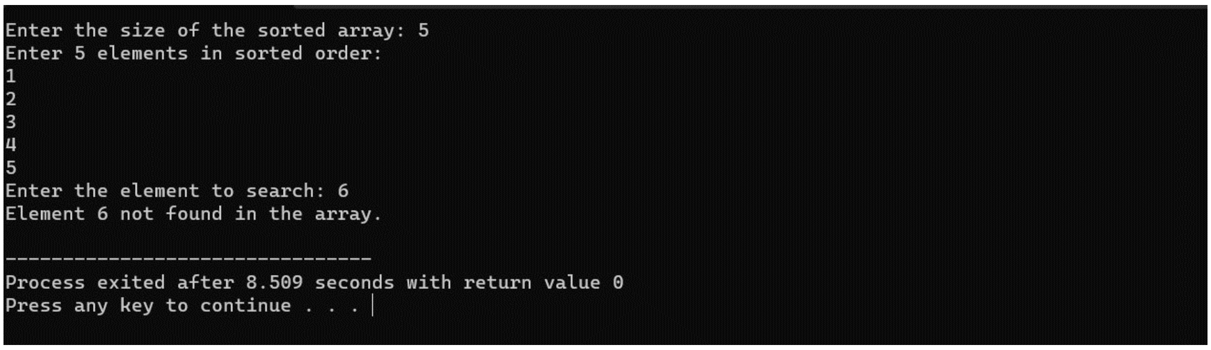
```
Enter the size of the array: 5
Enter 5 elements:
1
2
3
4
5
Enter the element to search: 4
Element 4 found at index 3.

--------------------------------
Process exited after 11.74 seconds with return value 0
Press any key to continue . . .
```

8.Write a c program to find element in an array using binary search?

Program:

```
#include <stdio.h>
int binarySearch(int array[], int size, int key) {
    int left = 0;
    int right = size - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (array[mid] == key) {
            return mid;
        } else if (array[mid] < key) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return -1;
}
int main() {
    int size, key ,i;
    printf("Enter the size of the sorted array: ");
    scanf("%d", &size);
    int array[size];
    printf("Enter %d elements in sorted order:\n", size);
    for ( i = 0; i < size; i++) {
        scanf("%d", &array[i]);
    }
    printf("Enter the element to search: ");
    scanf("%d", &key);
    int index = binarySearch(array, size, key);
    if (index != -1) {
        printf("Element %d found at index %d.\n", key, index);
    } else {
        printf("Element %d not found in the array.\n", key);
    }
    return 0;
}
```

Output:



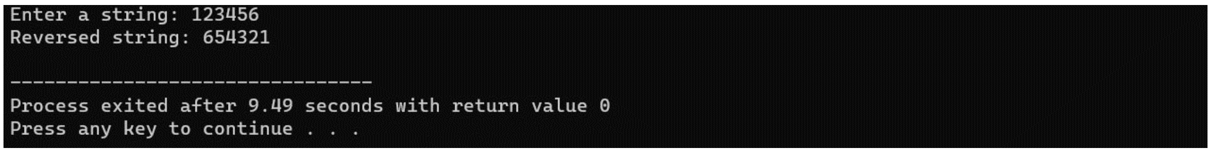
```
Enter the size of the sorted array: 5
Enter 5 elements in sorted order:
1
2
3
4
5
Enter the element to search: 6
Element 6 not found in the array.

--------------------------------
Process exited after 8.509 seconds with return value 0
Press any key to continue . . .
```

9.Write a c program to reverse a given String?

Program:

```
#include <stdio.h>

#include <string.h>

void reverseString(char str[]) {

   int length = strlen(str) ,i;

   for ( i = 0; i < length / 2; i++) {

      char temp = str[i];

      str[i] = str[length - 1 - i];

      str[length - 1 - i] = temp;

   }

}

int main() {

   char input[100];

   printf("Enter a string: ");

   scanf("%s", input);

   reverseString(input);

   printf("Reversed string: %s\n", input);

   return 0;

}
```

Output:



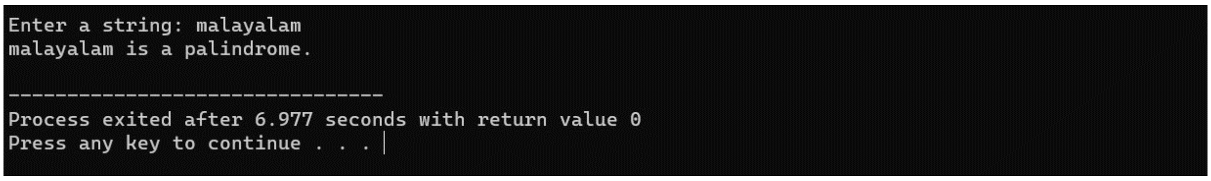
```
Enter a string: 123456
Reversed string: 654321

--------------------------------
Process exited after 9.49 seconds with return value 0
Press any key to continue . . .
```

10.Write a c program to find string is palindrome or not?

Program:

```
#include <stdio.h>

#include <string.h>

int isPalindrome(char str[]) {

   int length = strlen(str) ,i;

   for ( i = 0; i < length / 2; i++) {

      if (str[i] != str[length - 1 - i]) {
           return 0;
      }
   }
   return 1;
}
int main() {
   char input[100];
   printf("Enter a string: ");
   scanf("%s", input);
   if (isPalindrome(input)) {
      printf("%s is a palindrome.\n", input);
   } else {
      printf("%s is not a palindrome.\n", input);
   }
   return 0;
}
```

Output:



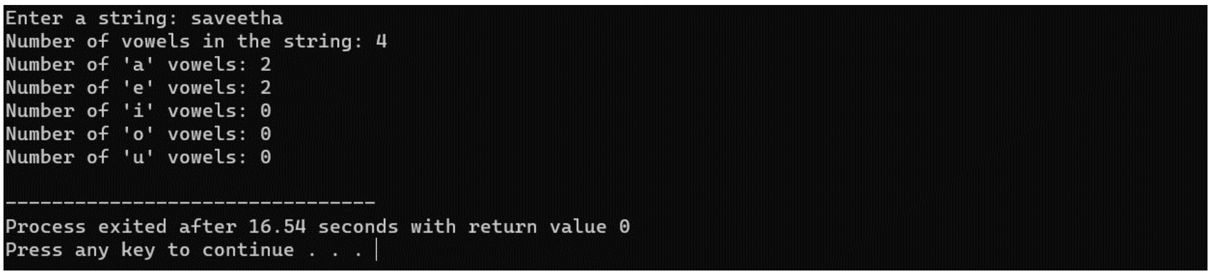
```
Enter a string: malayalam
malayalam is a palindrome.

--------------------------------
Process exited after 6.977 seconds with return value 0
Press any key to continue . . .
```

11.write a c program to find and count number of times vowels are present in given string?

Program:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int isVowel(char ch) {
    ch = tolower(ch);
    return (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u');
}
int main() {
    int i;

    char input[100];

    printf("Enter a string: ");

    scanf("%s", input);

    int vowelCount[5] = {0};

    for (i = 0; i < strlen(input); i++) {

        if (isVowel(input[i])) {

            switch (tolower(input[i])) {

                case 'a':

                    vowelCount[0]++;

                    break;

                case 'e':

                    vowelCount[1]++;

                    break;

                case 'i':

                    vowelCount[2]++;

                    break;

                case 'o':

                    vowelCount[3]++;

                    break;

                case 'u':

                    vowelCount[4]++;

                    break;

            }

        }

    }

    printf("Number of vowels in the string: %d\n", vowelCount[0] + vowelCount[1] + vowelCount[2] +
vowelCount[3] + vowelCount[4]);

    printf("Number of 'a' vowels: %d\n", vowelCount[0]);

    printf("Number of 'e' vowels: %d\n", vowelCount[1]);
    printf("Number of 'i' vowels: %d\n", vowelCount[2]);

    printf("Number of 'o' vowels: %d\n", vowelCount[3]);

    printf("Number of 'u' vowels: %d\n", vowelCount[4]);

    return 0;

}
```

Output:



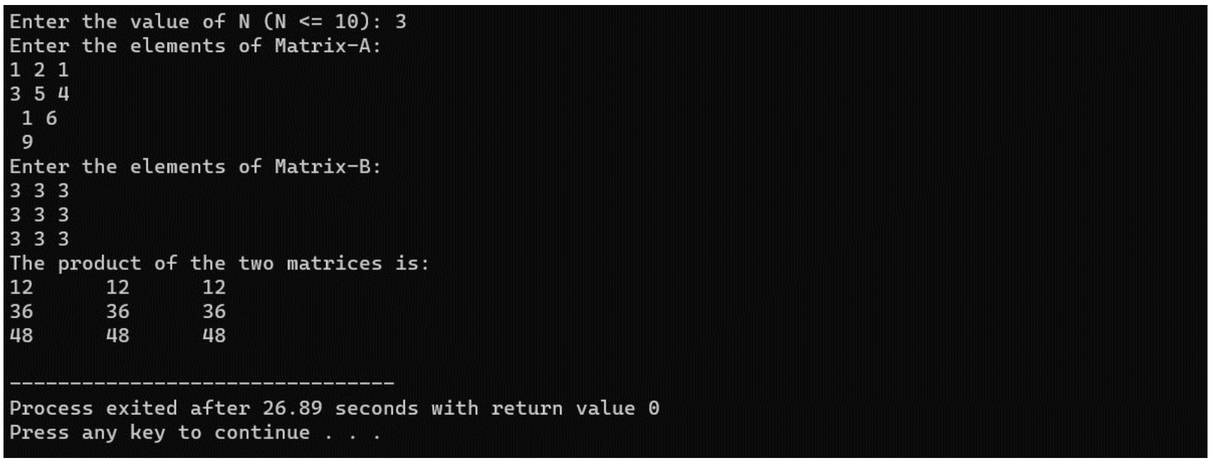
```
Enter a string: saveetha
Number of vowels in the string: 4
Number of 'a' vowels: 2
Number of 'e' vowels: 2
Number of 'i' vowels: 0
Number of 'o' vowels: 0
Number of 'u' vowels: 0

--------------------------------
Process exited after 16.54 seconds with return value 0
Press any key to continue . . .
```

12. write a c program for matrix multiplication?

Program:

```
#include<stdio.h>

int main() {

    int a[10][10], b[10][10], c[10][10], n, i, j, k;

    printf("Enter the value of N (N <= 10): ");

    scanf("%d", & n);

    printf("Enter the elements of Matrix-A: \n");

    for (i = 0; i < n; i++) {

        for (j = 0; j < n; j++) {

            scanf("%d", & a[i][j]);

        }

    }

    printf("Enter the elements of Matrix-B: \n");

    for (i = 0; i < n; i++) {

        for (j = 0; j < n; j++) {

            scanf("%d", & b[i][j]);

        }

    }
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            c[i][j] = 0;
            for (k = 0; k < n; k++) {
                c[i][j] += a[i][k] * b[k][j];
            }
        }
    }
    printf("The product of the two matrices is: \n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d\t", c[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Output:



```
Enter the value of N (N <= 10): 3
Enter the elements of Matrix-A:
1 2 1
3 5 4
 1 6
 9
Enter the elements of Matrix-B:
3 3 3
3 3 3
3 3 3
The product of the two matrices is:
12      12      12
36      36      36
48      48      48

--------------------------------
Process exited after 26.89 seconds with return value 0
Press any key to continue . . .
```

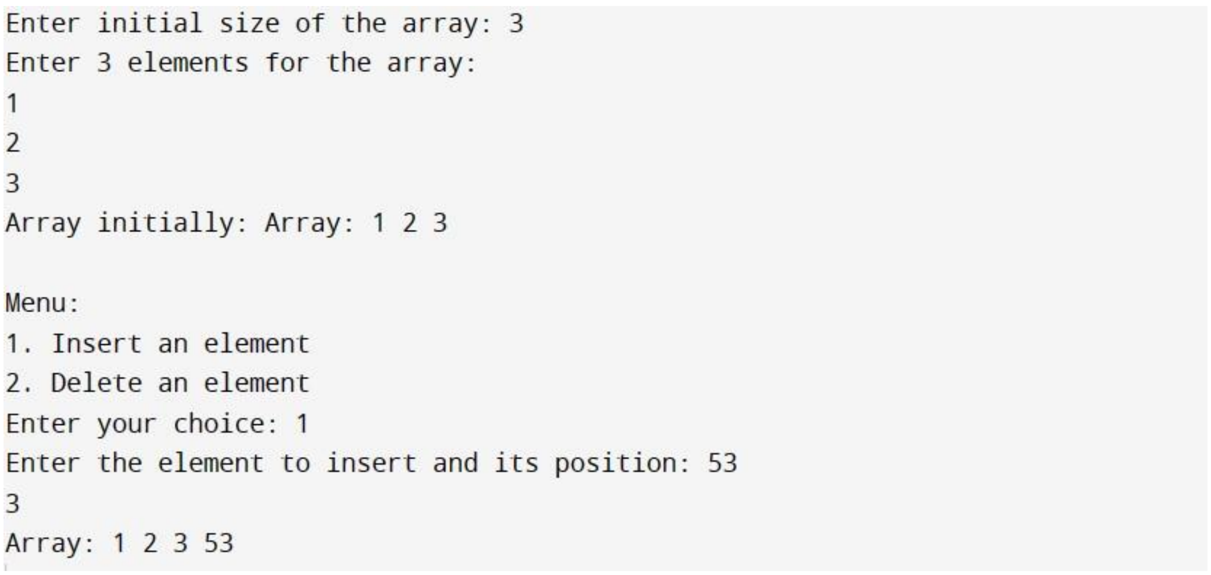
13.Write a c program to perform following operations into an array 1)Insert an element 2)delete an element?

Program:

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

```
#define MAX_SIZE 100

void displayArray(int arr[], int size) {

   printf("Array:");

   for (int i = 0; i < size; i++) printf(" %d", arr[i]);

   printf("\n");

}

void insertElement(int arr[], int *size, int position, int element) {

   if (*size >= MAX_SIZE || position < 0 || position > *size) {

      printf("Invalid operation!\n");

      return;

   }

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

   arr[position] = element;

   (*size)++;

   displayArray(arr, *size);

}

void deleteElement(int arr[], int *size, int position) {

   if (*size <= 0 || position < 0 || position >= *size) {

      printf("Invalid operation!\n");

      return;

   }

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

   (*size)--;

   displayArray(arr, *size);

}

int main() {

   int arr[MAX_SIZE], size;

   printf("Enter initial size of the array: ");

   scanf("%d", &size);

   if (size < 0 || size > MAX_SIZE) {
      printf("Invalid size!\n");
      return 1;
   }
   printf("Enter %d elements for the array:\n", size);
   for (int i = 0; i < size; i++) scanf("%d", &arr[i]);
   printf("\nArray initially: ");
   displayArray(arr, size);
   int choice, element, position;
   printf("\nMenu:\n1. Insert an element\n2. Delete an element\nEnter your choice: ");
   scanf("%d", &choice);
   switch (choice) {
      case 1:
          printf("Enter the element to insert and its position: ");
          scanf("%d %d", &element, &position);
          insertElement(arr, &size, position, element);
          break;
      case 2:
          printf("Enter the position to delete: ");
          scanf("%d", &position);
          deleteElement(arr, &size, position);
          break;
      default:
          printf("Invalid choice!\n");
   }
   return 0;
}
```

Output:



```
Enter initial size of the array: 3
Enter 3 elements for the array:
1
2
3
Array initially: Array: 1 2 3

Menu:
1. Insert an element
2. Delete an element
Enter your choice: 1
Enter the element to insert and its position: 53
3
Array: 1 2 3 53
```

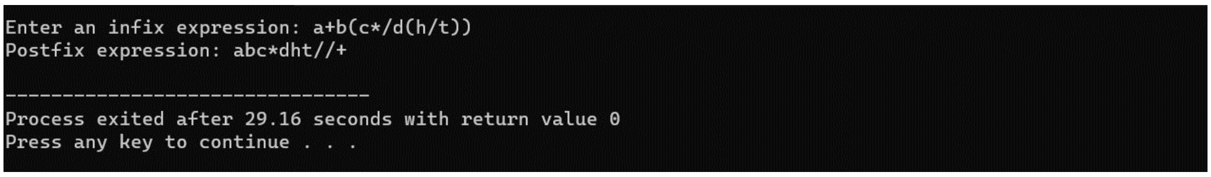
DAY 3

1.Write a c program for infix to postfix expression?

Program:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX_SIZE 100

struct Stack {

    char items[MAX_SIZE];

    int top;

};


void initialize(struct Stack *stack) {

    stack->top = -1;

}

int isEmpty(struct Stack *stack) {

    return stack->top == -1;

}

void push(struct Stack *stack, char item) {

    if (stack->top >= MAX_SIZE - 1) {

        printf("Stack is full. Cannot push.\n");
        return;
    }
    stack->items[++stack->top] = item;
}
char pop(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty. Cannot pop.\n");
        return '\0';
    }
    return stack->items[stack->top--];
}
int isOperator(char ch) {
    return ch == '+' || ch == '-' || ch == '*' || ch == '/';
}
int precedence(char ch) {
    if (ch == '+' || ch == '-')
        return 1;
    if (ch == '*' || ch == '/')
        return 2;
    return 0;
}
void infixToPostfix(char infix[], char postfix[]) {
    struct Stack stack;
    initialize(&stack);
    int postfixIndex = 0, i;
    for ( i = 0; infix[i] != '\0'; i++) {
        char ch = infix[i];
        if (ch == ' ')
            continue;
        if (isdigit(ch) || isalpha(ch)) {
            postfix[postfixIndex++] = ch;
        } else if (ch == '(') {
            push(&stack, ch);
        } else if (ch == ')') {
            while (!isEmpty(&stack) && stack.items[stack.top] != '(') {
                postfix[postfixIndex++] = pop(&stack);
            }
            pop(&stack);
        } else if (isOperator(ch)) {
            while (!isEmpty(&stack) && precedence(stack.items[stack.top]) >= precedence(ch)) {
                postfix[postfixIndex++] = pop(&stack);
            }
            push(&stack, ch);
        }
    }
    while (!isEmpty(&stack)) {
        postfix[postfixIndex++] = pop(&stack);
    }
    postfix[postfixIndex] = '\0';
}
int main() {
    char infix[MAX_SIZE], postfix[MAX_SIZE];
    printf("Enter an infix expression: ");
    gets(infix);
    infixToPostfix(infix, postfix);
    printf("Postfix expression: %s\n", postfix);
    return 0;
}
```

Output:



```
Enter an infix expression: a+b(c*/d(h/t))
Postfix expression: abc*dht//+

--------------------------------
Process exited after 29.16 seconds with return value 0
Press any key to continue . . .
```

2.Write a c program for queue data structure?

Program:

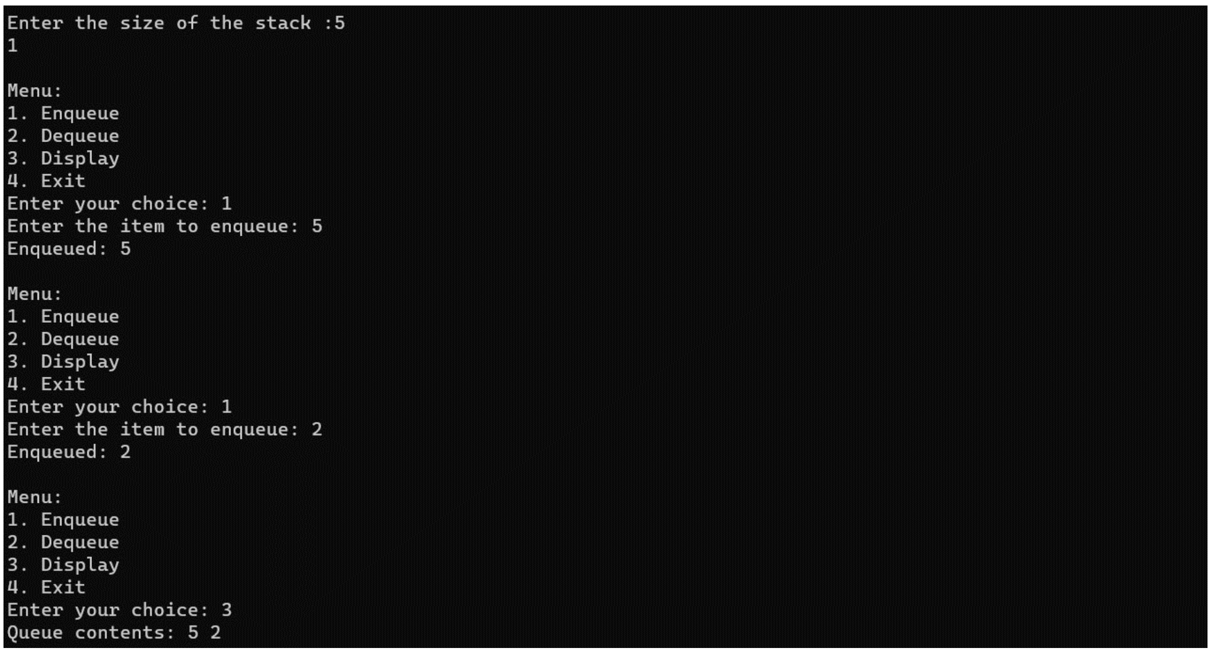
```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 100
struct Queue {
   int items[MAX_SIZE];
   int front;
   int rear;
};
void initialize(struct Queue *queue) {
   queue->front = -1;
   queue->rear = -1;
}
int isEmpty(struct Queue *queue) {
   return queue->front == -1;
}
int isFull(struct Queue *queue) {
   return (queue->rear + 1) % MAX_SIZE == queue->front;
}
void enqueue(struct Queue *queue, int item) {
   if (isFull(queue)) {
      printf("Queue is full. Cannot enqueue %d.\n", item);
      return;
   }
   if (isEmpty(queue)) {
      queue->front = 0;
      queue->rear = 0;
   } else {
      queue->rear = (queue->rear + 1) % MAX_SIZE;
   }
```

```
    queue->items[queue->rear] = item;
    printf("Enqueued: %d\n", item);
}
int dequeue(struct Queue *queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty. Cannot dequeue.\n");
        return -1;
    }
    int dequeuedItem = queue->items[queue->front];
    if (queue->front == queue->rear) {
        queue->front = -1;
        queue->rear = -1;
    } else {
        queue->front = (queue->front + 1) % MAX_SIZE;
    }
    printf("Dequeued: %d\n", dequeuedItem);
    return dequeuedItem;
}
void display(struct Queue *queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty.\n");
        return;
    }
    printf("Queue contents:");
    int i = queue->front;
    while (i != queue->rear) {
        printf(" %d", queue->items[i]);
        i = (i + 1) % MAX_SIZE;
    }
    printf(" %d", queue->items[i]);
    printf("\n");
```

```
}
int main() {
    struct Queue queue;
    initialize(&queue);
    int choice, item, n, i;
    printf("Enter the size of the stack :");
    scanf("%d",&n);
    for(i = 0;i < n;i++){
             scanf("%d",&n);
             }
    do {
        printf("\nMenu:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the item to enqueue: ");
                scanf("%d", &item);
                enqueue(&queue, item);
                break;
            case 2:
                dequeue(&queue);
                break;
            case 3:
                display(&queue);
                break;
            case 4:
```

```
            printf("Exiting...\n");
            break;
          default:
            printf("Invalid choice!\n");
        }
    } while (choice != 4);
    return 0;
}
```

Output:



```
Enter the size of the stack :5
1

Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the item to enqueue: 5
Enqueued: 5

Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the item to enqueue: 2
Enqueued: 2

Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Queue contents: 5 2
```

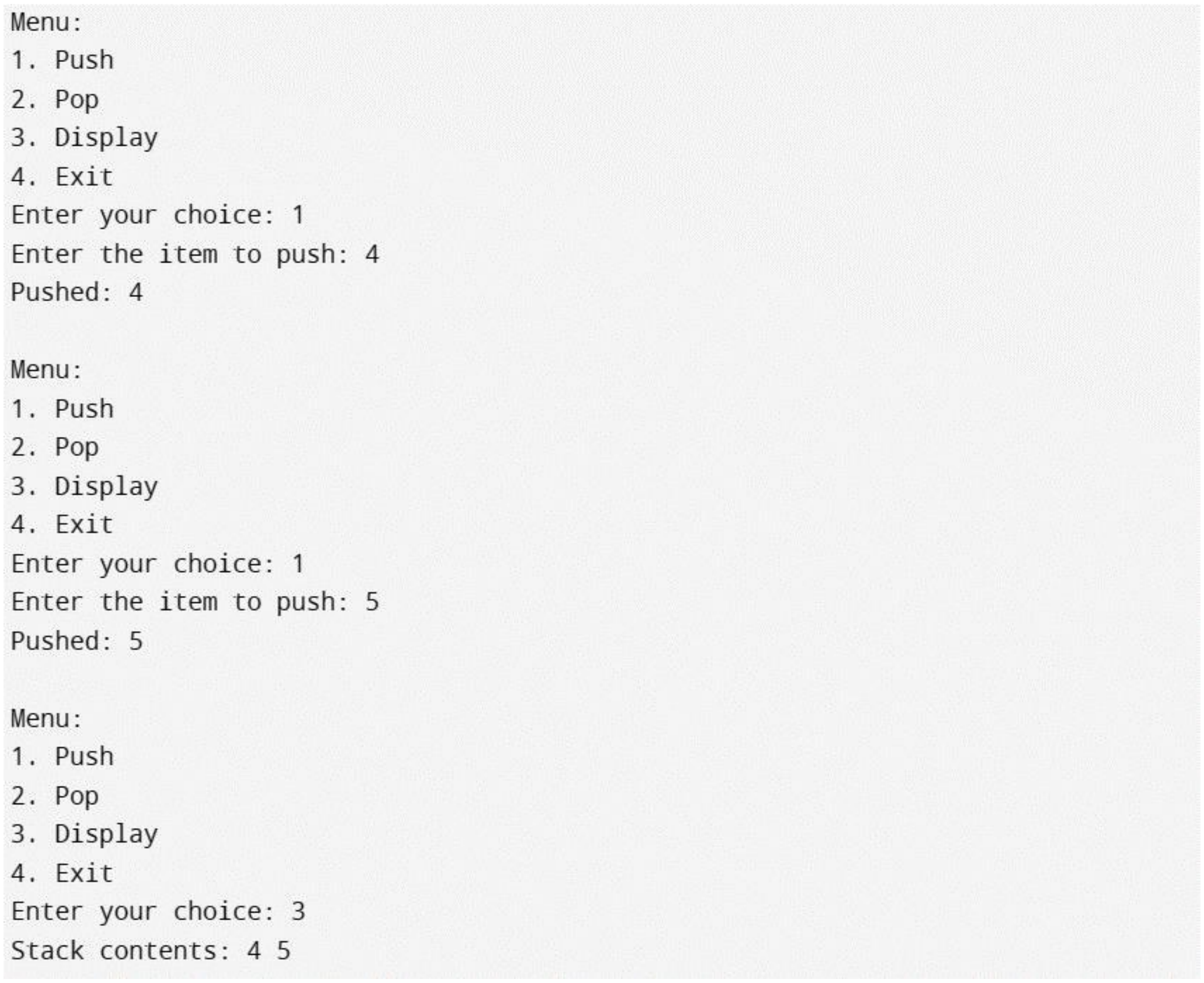
3.Write a c program for to implement stack operations?

Program:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 100
struct Stack {
    int items[MAX_SIZE];
    int top;
};
void initialize(struct Stack *stack) {
    stack->top = -1;
```

```
}
int isEmpty(struct Stack *stack) {
    return stack->top == -1;
}
int isFull(struct Stack *stack) {
    return stack->top == MAX_SIZE - 1;
}
void push(struct Stack *stack, int item) {
    if (isFull(stack)) {
        printf("Stack is full. Cannot push %d.\n", item);
        return;
    }
    stack->items[++stack->top] = item;
    printf("Pushed: %d\n", item);
}
int pop(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty. Cannot pop.\n");
        return -1;
    }
    int poppedItem = stack->items[stack->top--];
    printf("Popped: %d\n", poppedItem);
    return poppedItem;
}
void display(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty.\n");
        return;
    }
    printf("Stack contents:");
    for (int i = 0; i <= stack->top; i++) {
        printf(" %d", stack->items[i]);
    }
    printf("\n");
}
int main() {
    struct Stack stack;
    initialize(&stack);
    int choice, item;
    do {
        printf("\nMenu:\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the item to push: ");
                scanf("%d", &item);
                push(&stack, item);
                break;
            case 2:
                pop(&stack);
                break;
            case 3:
                display(&stack);
                break;
            case 4:
                printf("Exiting...\n");
                break;
        default:
            printf("Invalid choice!\n");
        }
    } while (choice != 4);
    return 0;
}
```

Output:



```
Menu:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the item to push: 4
Pushed: 4

Menu:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the item to push: 5
Pushed: 5

Menu:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack contents: 4 5
```

4.Write a c program to implement linked list?

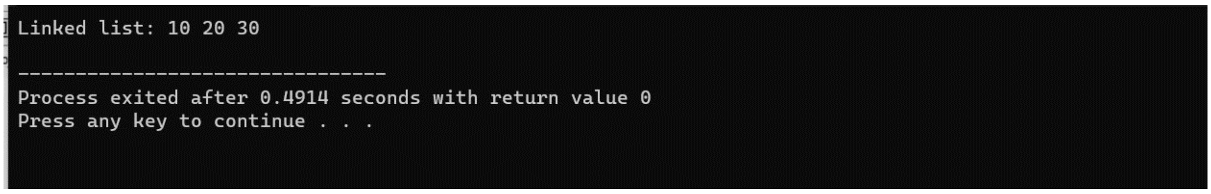
Program :

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
```

```
};
struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}
void insertEnd(struct Node** head, int data) {
   struct Node* newNode = createNode(data);
   if (*head == NULL) {
      *head = newNode;
      return;
   }
   struct Node* current = *head;
   while (current->next != NULL) {
      current = current->next;
   }
   current->next = newNode;
}
void displayList(struct Node* head) {
   struct Node* current = head;
   while (current != NULL) {
      printf("%d ", current->data);
      current = current->next;
   }
   printf("\n");
}
int main() {
   struct Node* head = NULL;
   insertEnd(&head, 10);
   insertEnd(&head, 20);
```

```
    insertEnd(&head, 30);
    printf("Linked list: ");
    displayList(head);
    return 0;
}
```

Output:



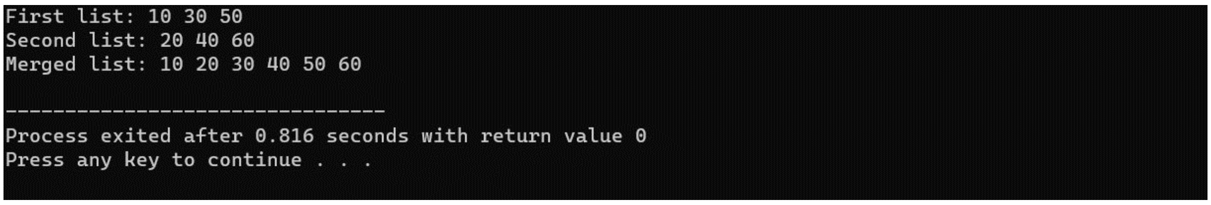
```
Linked list: 10 20 30

--------------------------------
Process exited after 0.4914 seconds with return value 0
Press any key to continue . . .
```

5.Write a c program for merge two list?

Program:

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
void insertEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node* current = *head;
    while (current->next != NULL) {
        current = current->next;
    }
    current->next = newNode;
}
struct Node* mergeLists(struct Node* list1, struct Node* list2) {
    if (list1 == NULL) return list2;
    if (list2 == NULL) return list1;
    struct Node* result = NULL;
    if (list1->data <= list2->data) {
        result = list1;
        result->next = mergeLists(list1->next, list2);
    } else {
        result = list2;
        result->next = mergeLists(list1, list2->next);
    }
    return result;
}
void displayList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}
int main() {
    struct Node* list1 = NULL;
    struct Node* list2 = NULL;
    insertEnd(&list1, 10);
    insertEnd(&list1, 30);
    insertEnd(&list1, 50);

    insertEnd(&list2, 20);

    insertEnd(&list2, 40);

    insertEnd(&list2, 60);

    printf("First list: ");

    displayList(list1);


    printf("Second list: ");

    displayList(list2);

    struct Node* mergedList = mergeLists(list1, list2);

    printf("Merged list: ");

    displayList(mergedList);


    return 0;

}
```

Output:



```
First list: 10 30 50
Second list: 20 40 60
Merged list: 10 20 30 40 50 60

--------------------------------
Process exited after 0.816 seconds with return value 0
Press any key to continue . . .
```

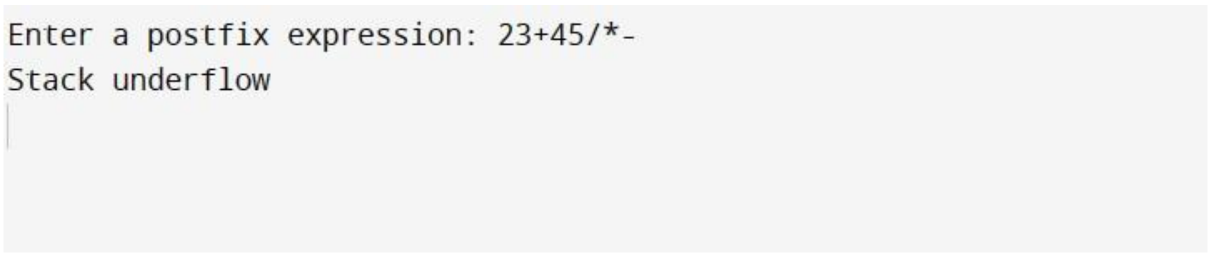
6.Write a c program to evaluate the postfix expression?

Program:

```
#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#define MAX_STACK_SIZE 100

typedef struct {

    int data[MAX_STACK_SIZE];

    int top;

} Stack;
```

```
void initialize(Stack *s) {
    s->top = -1;
}
void push(Stack *s, int value) {
    if (s->top < MAX_STACK_SIZE - 1) {
        s->top++;
        s->data[s->top] = value;
    } else {
        printf("Stack overflow\n");
        exit(1);
    }
}
int pop(Stack *s) {
    if (s->top >= 0) {
        int value = s->data[s->top];
        s->top--;
        return value;
    } else {
        printf("Stack underflow\n");
        exit(1);
    }
}
int evaluatePostfix(char postfix[]) {
    Stack stack;
    initialize(&stack);
    for (int i = 0; postfix[i] != '\0'; i++) {
        if (isdigit(postfix[i])) {
            push(&stack, postfix[i] - '0');
        } else {
            int operand2 = pop(&stack);
            int operand1 = pop(&stack);
            switch (postfix[i]) {
              case '+':
                push(&stack, operand1 + operand2);
                break;
              case '-':
                push(&stack, operand1 - operand2);
                break;
              case '*':
                push(&stack, operand1 * operand2);
                break;
              case '/':
                push(&stack, operand1 / operand2);
                break;
              default:
                printf("Invalid operator\n");
                exit(1);
            }
        }
    }
    return pop(&stack);
}
int main() {
    char postfix[100];
    printf("Enter a postfix expression: ");
    scanf("%s", postfix);
    int result = evaluatePostfix(postfix);
    printf("Result: %d\n", result);
    return 0;
}
```

Output:


```
Enter a postfix expression: 23+45/*-
Stack underflow
```

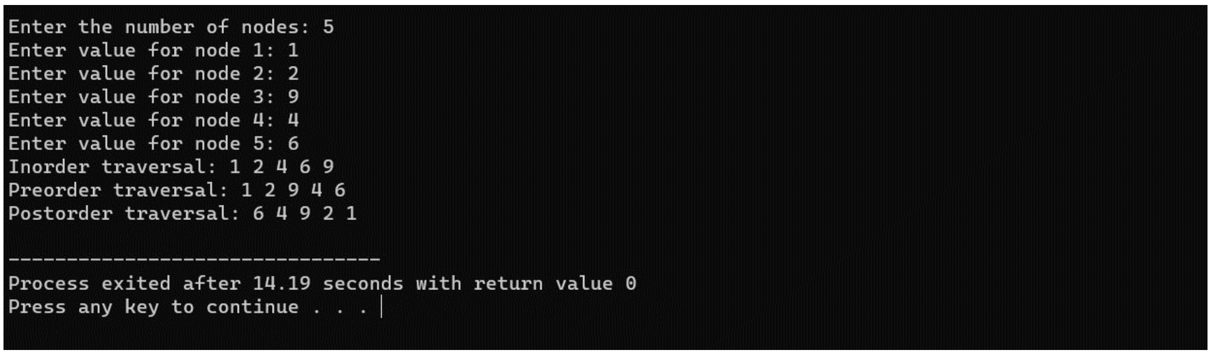
7.write a c program to implement tree traversals?

Program:

```
#include <stdio.h>

#include <stdlib.h>

struct Node {

   int data;

   struct Node *left;

   struct Node *right;

};

struct Node *newNode(int data) {

   struct Node *node = (struct Node *)malloc(sizeof(struct Node));

   node->data = data;

   node->left = NULL;

   node->right = NULL;

   return node;

}

void inorderTraversal(struct Node *root) {

   if (root != NULL) {

      inorderTraversal(root->left);

      printf("%d ", root->data);

      inorderTraversal(root->right);

   }

}

void preorderTraversal(struct Node *root) {

   if (root != NULL) {

      printf("%d ", root->data);
```

```
        preorderTraversal(root->left);
        preorderTraversal(root->right);
    }
}
void postorderTraversal(struct Node *root) {
    if (root != NULL) {
        postorderTraversal(root->left);
        postorderTraversal(root->right);
        printf("%d ", root->data);
    }
}
int main() {
    struct Node *root = NULL;
    int n, i;
    printf("Enter the number of nodes: ");
    scanf("%d", &n);
    for ( i = 0; i < n; i++) {
        int value;
        printf("Enter value for node %d: ", i + 1);
        scanf("%d", &value);
        if (root == NULL) {
            root = newNode(value);
        } else {
            struct Node *current = root;
            struct Node *parent = NULL;
            while (current != NULL) {
                parent = current;
                if (value < current->data) {
                    current = current->left;
                } else {
                    current = current->right;
                }
            }
            if (value < parent->data) {
                parent->left = newNode(value);
            } else {
                parent->right = newNode(value);
            }
        }
    }
    printf("Inorder traversal: ");
    inorderTraversal(root);
    printf("\n");
    printf("Preorder traversal: ");
    preorderTraversal(root);
    printf("\n");
    printf("Postorder traversal: ");
    postorderTraversal(root);
    printf("\n");
    return 0;
}
```

Output:



```
Enter the number of nodes: 5
Enter value for node 1: 1
Enter value for node 2: 2
Enter value for node 3: 9
Enter value for node 4: 4
Enter value for node 5: 6
Inorder traversal: 1 2 4 6 9
Preorder traversal: 1 2 9 4 6
Postorder traversal: 6 4 9 2 1

--------------------------------
Process exited after 14.19 seconds with return value 0
Press any key to continue . . .
```

DAY 4

1.Write a c program to implement binary search tree?

Program:

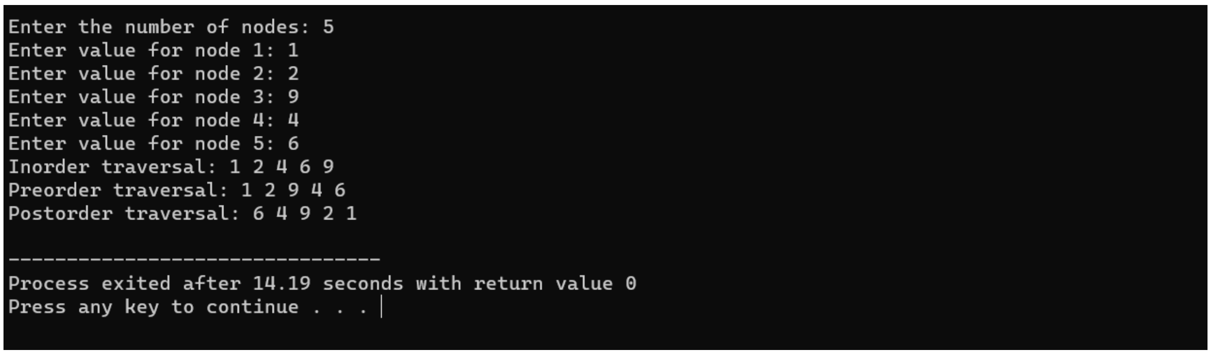
```
#include <stdio.h>
```

```
#include <stdlib.h>

struct Node {

   int data;

   struct Node *left;

   struct Node *right;

};

struct Node *newNode(int data) {

   struct Node *node = (struct Node *)malloc(sizeof(struct Node));

   node->data = data;

   node->left = NULL;

   node->right = NULL;

   return node;

}

struct Node *insert(struct Node *root, int data) {

   if (root == NULL) {

      return newNode(data);

   }

   if (data < root->data) {

      root->left = insert(root->left, data);

   } else if (data > root->data) {

      root->right = insert(root->right, data);

   }

   return root;

}

void inorderTraversal(struct Node *root) {

   if (root != NULL) {

      inorderTraversal(root->left);

      printf("%d ", root->data);

      inorderTraversal(root->right);

   }
```

```
}
void preorderTraversal(struct Node *root) {
  if (root != NULL) {
    printf("%d ", root->data);
    preorderTraversal(root->left);
    preorderTraversal(root->right);
  }
}
void postorderTraversal(struct Node *root) {
  if (root != NULL) {
    postorderTraversal(root->left);
    postorderTraversal(root->right);
    printf("%d ", root->data);
  }
}
int main() {
  struct Node *root = NULL;

  int n, i;
  printf("Enter the number of nodes: ");
  scanf("%d", &n);
  printf("Enter the values:\n");
  for ( i = 0; i < n; i++) {
    int value;
    scanf("%d", &value);
    root = insert(root, value);
  }
  printf("In-order traversal: ");
  inorderTraversal(root);
  printf("\n");
```

```
    printf("Pre-order traversal: ");

    preorderTraversal(root);

    printf("\n");

    printf("Post-order traversal: ");

    postorderTraversal(root);

    printf("\n");

    return 0;

}
```

Output :



```
Enter the number of nodes: 5
Enter value for node 1: 1
Enter value for node 2: 2
Enter value for node 3: 9
Enter value for node 4: 4
Enter value for node 5: 6
Inorder traversal: 1 2 4 6 9
Preorder traversal: 1 2 9 4 6
Postorder traversal: 6 4 9 2 1

--------------------------------
Process exited after 14.19 seconds with return value 0
Press any key to continue . . .
```

2.Write a C program to implement AVL Tree?

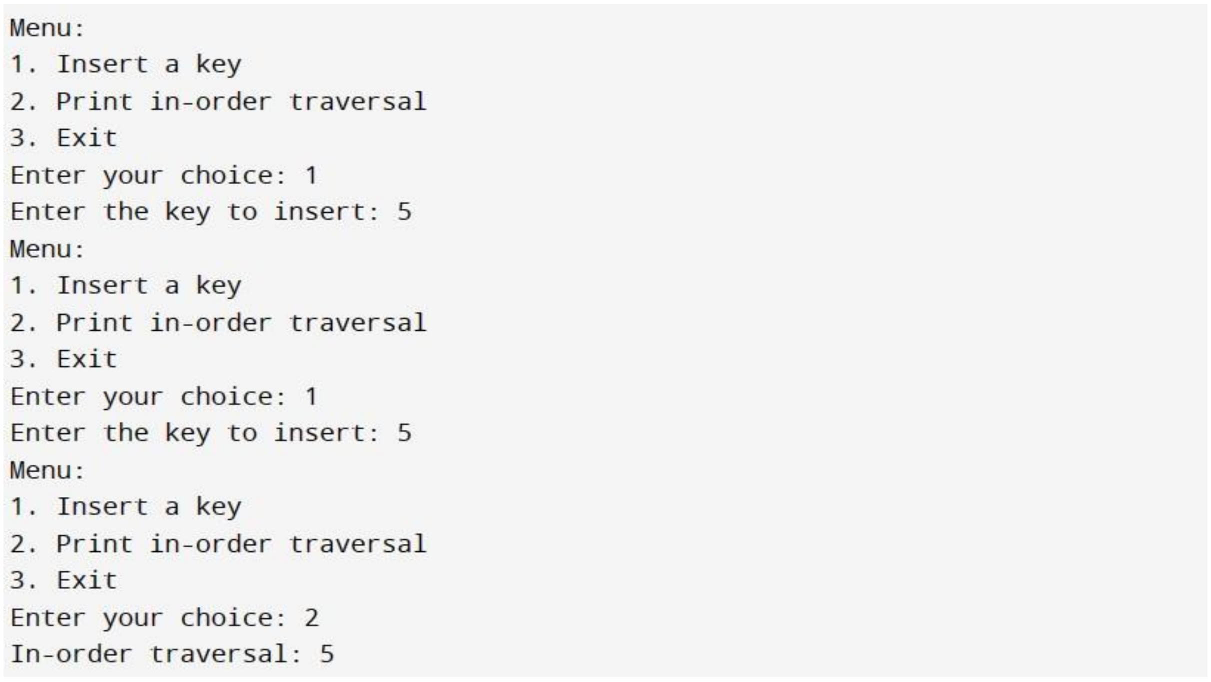
Program:

```
#include <stdio.h>

#include <stdlib.h>

struct Node {

    int key;

    struct Node* left;

    struct Node* right;

    int height;

};

struct Node* newNode(int key) {

    struct Node* node = (struct Node*)malloc(sizeof(struct Node));

    node->key = key;

    node->left = node->right = NULL;

    node->height = 1;
    return node;
}
int height(struct Node* node) {
    if (node == NULL)
        return 0;
    return node->height;
}
int max(int a, int b) {
    return (a > b) ? a : b;
}
struct Node* rightRotate(struct Node* y) {
    struct Node* x = y->left;
    struct Node* T2 = x->right;
    x->right = y;
    y->left = T2;
    y->height = max(height(y->left), height(y->right)) + 1;
    x->height = max(height(x->left), height(x->right)) + 1;
    return x;
}
struct Node* leftRotate(struct Node* x) {
    struct Node* y = x->right;
    struct Node* T2 = y->left;
    y->left = x;
    x->right = T2;
    x->height = max(height(x->left), height(x->right)) + 1;
    y->height = max(height(y->left), height(y->right)) + 1;
    return y;
}
int getBalance(struct Node* node) {
    if (node == NULL)
```

```
        return 0;
    return height(node->left) - height(node->right);
}
struct Node* insert(struct Node* node, int key) {
    if (node == NULL)
        return newNode(key);
    if (key < node->key)
        node->left = insert(node->left, key);
    else if (key > node->key)
                return node;
    node->height = 1 + max(height(node->left), height(node->right));
    int balance = getBalance(node);
    if (balance > 1) {
        if (key < node->left->key) {
            return rightRotate(node);
        } else {
            node->left = leftRotate(node->left);
            return rightRotate(node);
        }
    }
    if (balance < -1) {
        if (key > node->right->key) {
            return leftRotate(node);
        } else {
            node->right = rightRotate(node->right);
            return leftRotate(node);
        }
    }
    return node;
}
```

```
void inOrder(struct Node* root) {
    if (root != NULL) {
        inOrder(root->left);
        printf("%d ", root->key);
        inOrder(root->right);
    }
}
int main() {
    struct Node* root = NULL;
    int choice, key;
    while (1) {
        printf("Menu:\n");
        printf("1. Insert a key\n");
        printf("2. Print in-order traversal\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the key to insert: ");
                scanf("%d", &key);
                root = insert(root, key);
                break;
            case 2:
                printf("In-order traversal: ");
                inOrder(root);
                printf("\n");
                break;
            case 3:
                exit(0);
            default:
                printf("Invalid choice!\n");
        }
    }
    return 0;
}
```

Output:



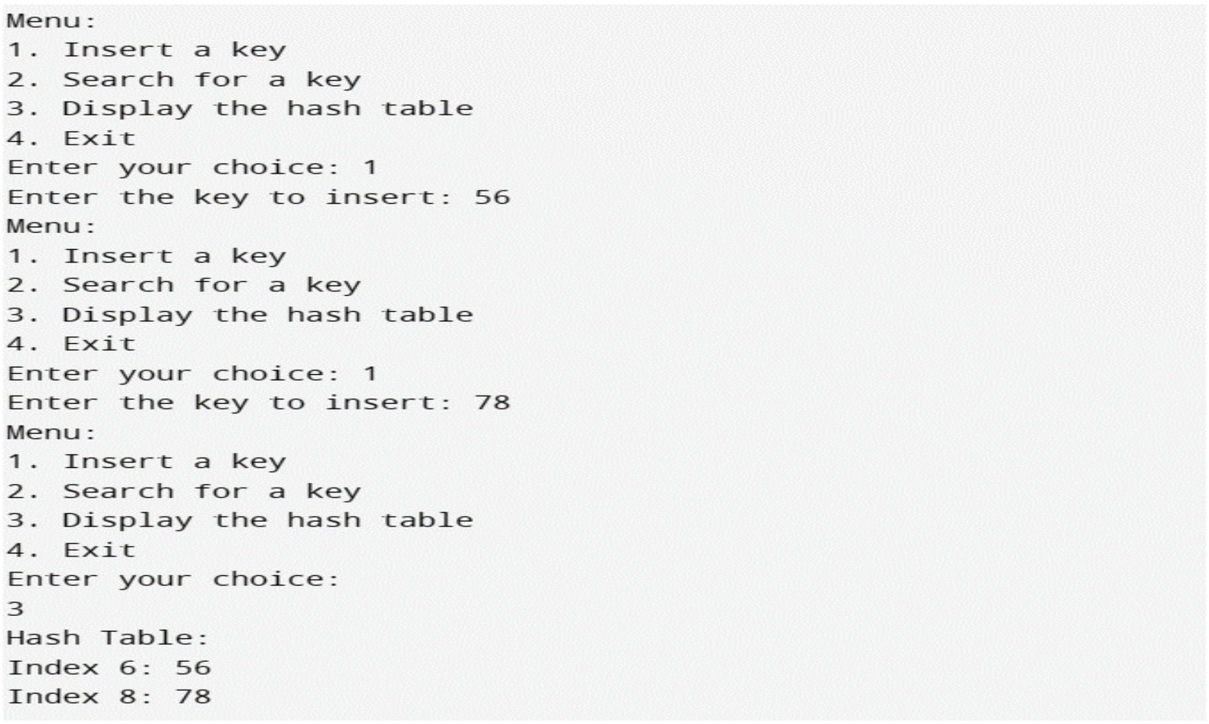
```
Menu:
1. Insert a key
2. Print in-order traversal
3. Exit
Enter your choice: 1
Enter the key to insert: 5
Menu:
1. Insert a key
2. Print in-order traversal
3. Exit
Enter your choice: 1
Enter the key to insert: 5
Menu:
1. Insert a key
2. Print in-order traversal
3. Exit
Enter your choice: 2
In-order traversal: 5
```

3.Write a C program to implement hashing using linear probing?

Program:

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10
struct HashTable {
    int table[SIZE];
    int count;
};
void initialize(struct HashTable* ht) {
    for (int i = 0; i < SIZE; i++) {
        ht->table[i] = -1;
    }
    ht->count = 0;
}
int hash(int key) {
    return key % SIZE;
}
void insert(struct HashTable* ht, int key) {
    if (ht->count == SIZE) {
        printf("Hash table is full. Cannot insert %d.\n", key);
        return;
    }
    int index = hash(key);
    while (ht->table[index] != -1) {
        index = (index + 1) % SIZE;
    }
    ht->table[index] = key;
    ht->count++;
}
int search(struct HashTable* ht, int key) {
    int index = hash(key);
    while (ht->table[index] != -1) {
        if (ht->table[index] == key) {
            return index;
        }
        index = (index + 1) % SIZE;
    }
    return -1;
}
void display(struct HashTable* ht) {
    printf("Hash Table:\n");
    for (int i = 0; i < SIZE; i++) {
        if (ht->table[i] != -1) {
            printf("Index %d: %d\n", i, ht->table[i]);
        }
    }
}
int main() {
    struct HashTable ht;
    initialize(&ht);
    int choice, key;
    do {
        printf("\nMenu:\n");
        printf("1. Insert a key\n");
        printf("2. Search for a key\n");
        printf("3. Display the hash table\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the key to insert: ");
                scanf("%d", &key);
                insert(&ht, key);
                break;
            case 2:
                printf("Enter the key to search: ");
                scanf("%d", &key);
                int index = search(&ht, key);
                if (index != -1) {
                    printf("Key %d found at index %d.\n", key, index);
                } else {
                    printf("Key %d not found.\n", key);
                }
                break;
            case 3:
                display(&ht);
                break;
            case 4:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice!\n");
        }
    } while (choice != 4);
    return 0;
}
```

Output:



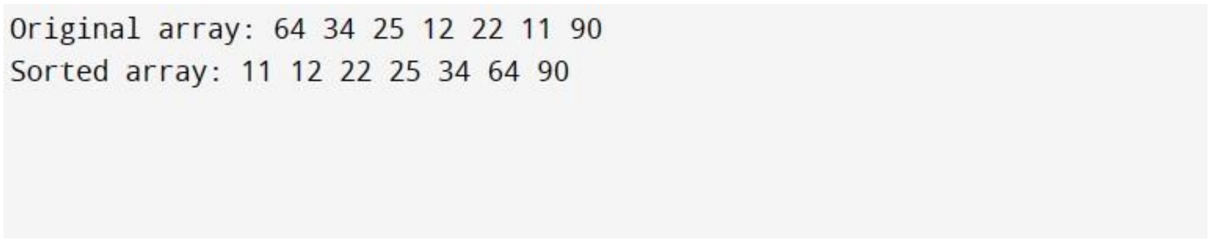
```
Menu:
1. Insert a key
2. Search for a key
3. Display the hash table
4. Exit
Enter your choice: 1
Enter the key to insert: 56
Menu:
1. Insert a key
2. Search for a key
3. Display the hash table
4. Exit
Enter your choice: 1
Enter the key to insert: 78
Menu:
1. Insert a key
2. Search for a key
3. Display the hash table
4. Exit
Enter your choice:
3
Hash Table:
Index 6: 56
Index 8: 78
```

4.Write a C program to implement bubble sort?

Program:

```
#include <stdio.h>

void bubbleSort(int arr[], int n) {

    int temp;

    int swapped;

    for (int i = 0; i < n - 1; i++) {

        swapped = 0;

        for (int j = 0; j < n - i - 1; j++) {

            if (arr[j] > arr[j + 1]) {

                temp = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = temp;

                swapped = 1;

            }

        }

        if (swapped == 0) {

            break;

        }

    }

}

int main() {

    int arr[] = {64, 34, 25, 12, 22, 11, 90};

    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: ");

    for (int i = 0; i < n; i++) {

        printf("%d ", arr[i]);

    }

    printf("\n");

    bubbleSort(arr, n);

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

Output:



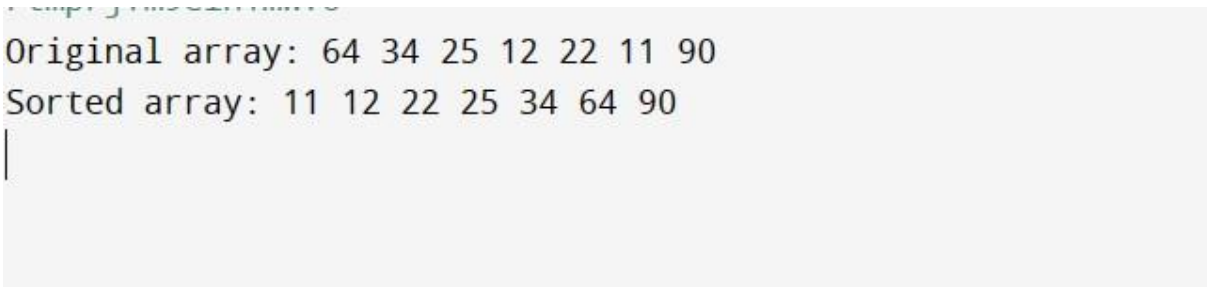
```
Original array: 64 34 25 12 22 11 90
Sorted array: 11 12 22 25 34 64 90
```

5.Write a C program to implement insertion sort?

Program:

```
#include <stdio.h>
void insertionSort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Original array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    insertionSort(arr, n);
    printf("Sorted array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

Output:



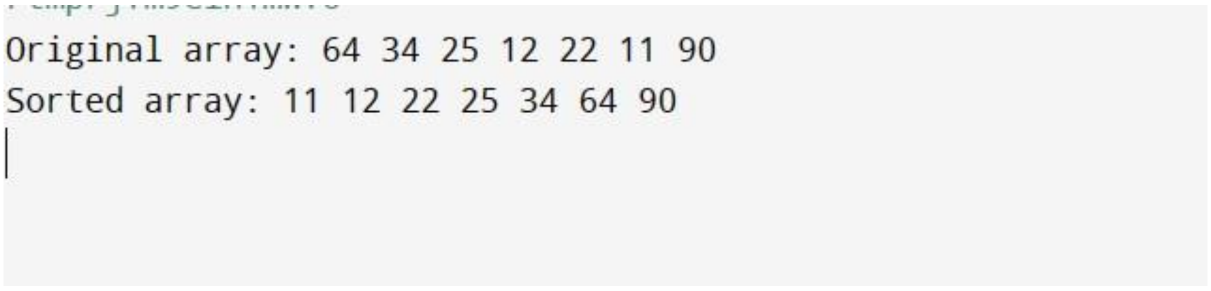
```
Original array: 64 34 25 12 22 11 90
Sorted array: 11 12 22 25 34 64 90
```

6.Write a c program to implement selection sort?

Program:

```
#include <stdio.h>
void selectionSort(int arr[], int n) {
    int i, j, minIndex, temp;
    for (i = 0; i < n - 1; i++) {
        minIndex = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
        temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
}
int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Original array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    selectionSort(arr, n);
    printf("Sorted array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

Output:



```
Original array: 64 34 25 12 22 11 90
Sorted array: 11 12 22 25 34 64 90
```

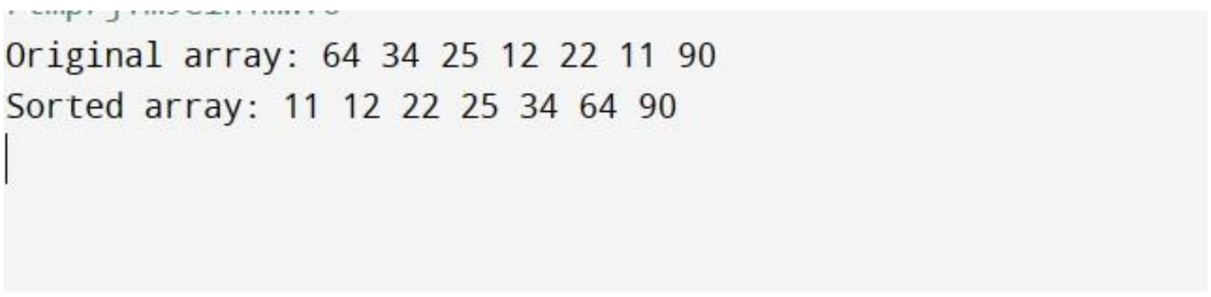
7.Write a c program to implement Quick sort?

Program:

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

```
void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);
    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pivotIndex = partition(arr, low, high);
        quickSort(arr, low, pivotIndex - 1);
        quickSort(arr, pivotIndex + 1, high);
    }
}
int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Original array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    quickSort(arr, 0, n - 1);
    printf("Sorted array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

Output:



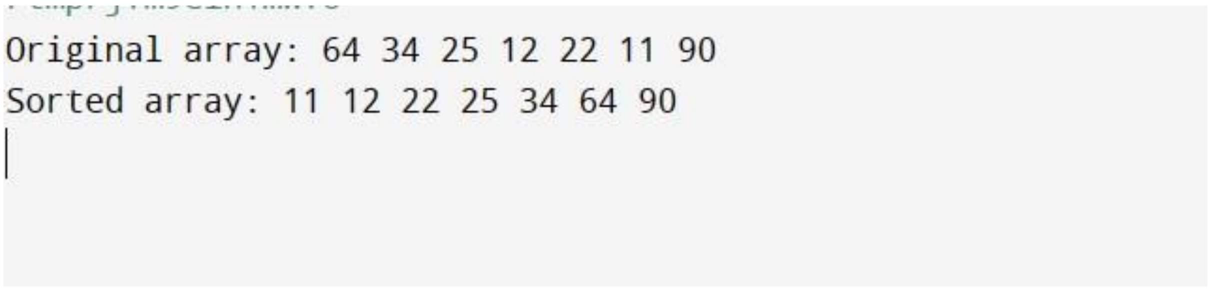
```
Original array: 64 34 25 12 22 11 90
Sorted array: 11 12 22 25 34 64 90
```

8.Write a C program to implement Merge sort?

Program:

```
#include <stdio.h>
void merge(int arr[], int l, int m, int r) {
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m
    int L[n1], R[n2];
    for (i = 0; i < n1; i++) {
        L[i] = arr[l + i];
    }
    for (j = 0; j < n2; j++) {
        R[j] = arr[m + 1 + j];
    }
    i = 0;

    j = 0;

    k = l;

    while (i < n1 && j < n2) {

        if (L[i] <= R[j]) {

            arr[k] = L[i];

            i++;

        } else {

            arr[k] = R[j];

            j++;

        }

        k++;

    }

    while (i < n1) {

        arr[k] = L[i];

        i++;

        k++;

    }

    while (j < n2) {

        arr[k] = R[j];

        j++;

        k++;

    }

}

void mergeSort(int arr[], int l, int r) {

    if (l < r) {

        int m = l + (r - l) / 2;

        mergeSort(arr, l, m);

        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}
int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Original array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    mergeSort(arr, 0, n - 1);
    printf("Sorted array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

Output:



```
Original array: 64 34 25 12 22 11 90
Sorted array: 11 12 22 25 34 64 90
```