

EXP-01

```
#include <stdio.h>

#include <time.h>

int square(int x)
{
    int s;

    s = x * x;

    return s;
}

int main()
{
    int i, sum = 0, temp;

    clock_t start, end;

    double cpu_time_used;

    // Start the clock

    start = clock();

    for(i = 1; i <= 10; i++)
    {
        temp = square(i);

        if(temp % 2 == 0)
        {
            sum = sum + temp;
        }

        else
        {
            sum = sum + (temp * 2);
        }
    }
}
```

```
    }
}

// End the clock

end = clock();

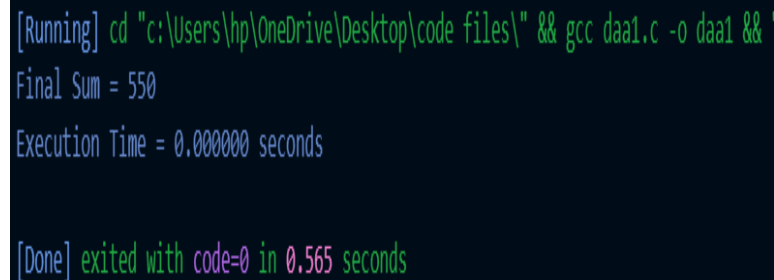
cpu_time_used = ((double)(end - start))
/ CLOCKS_PER_SEC;

printf("Final Sum = %d\n", sum);

printf("Execution Time = %f seconds\n",
cpu_time_used);

return 0;
}
```

OUTPUT:



```
[Running] cd "c:\Users\hp\OneDrive\Desktop\code files\" && gcc daa1.c -o daa1 && "
Final Sum = 550
Execution Time = 0.000000 seconds

[Done] exited with code=0 in 0.565 seconds
```

EXP-3

```
#include <stdio.h>

void merge(int a[], int low, int mid, int high) {

    int b[100];

    int i = low, j = mid + 1, k = low;

    while (i <= mid && j <= high) {

        if (a[i] <= a[j]) {

            b[k] = a[i];

            i++;

        } else {

            b[k] = a[j];

            j++;

        } k++; }

    while (i <= mid) {

        b[k] = a[i];

        i++;

        k++; }

    while (j <= high) {

        b[k] = a[j];

        j++;

        k++;

    }

    for (i = low; i <= high; i++) {

        a[i] = b[i];

    }

}
```

```
void mergeSort(int a[], int low, int high) {

    int mid;

    if (low < high) {

        mid = (low + high) / 2;

        mergeSort(a, low, mid);

        mergeSort(a, mid + 1, high);

        merge(a, low, mid, high);

    }

}

int main() {

    int n, a[100];

    printf("Enter number of elements: ");

    scanf("%d", &n);

    printf("Enter elements:\n");

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

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

    }

    mergeSort(a, 0, n - 1);

    printf("Sorted elements:\n");

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

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

    } return 0;}

}
```

OUTPUT:

```
PS C:\Users\hp\OneDrive\Desktop\code files> g++ -o code daa2.c
PS C:\Users\hp\OneDrive\Desktop\code files> .\daa2
Enter number of elements: 5
Enter elements:
15 40 36 32 20
Sorted elements:
15 20 32 36 40
```

EXP 2:

```
#include <stdio.h>

int binarySearch(int arr[], int n, int key) {

    int low = 0, high = n - 1, mid;

    while (low <= high) {

        mid = (low + high) / 2;

        if (arr[mid] == key) {

            return mid;

        }

        else if (arr[mid] < key) {

            low = mid + 1;

        }

        else {

            high = mid - 1;

        }

    }

    return -1;

}

int main() {

    int n, key;

    printf("Enter number of elements: ");

    scanf("%d", &n);

    int arr[100];
```

```
    printf("Enter elements in sorted
order:\n");

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

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

    }

    printf("Enter element to search: ");

    scanf("%d", &key);

    int result = binarySearch(arr, n, key);

    if (result != -1)

        printf("Element found at
position %d\n", result + 1);

    else

        printf("Element not found\n");

    return 0;

}
```

OUTPUT:

```
PS C:\Users\hp\OneDrive\Desktop\code files> g++ -o code da
PS C:\Users\hp\OneDrive\Desktop\code files> .\daa3
Enter number of elements: 6
Enter elements in sorted order:
20 30 45 65 70 80
Enter element to search: 65
Element found at position 4
```