Sure! Let's write a few classic **recursive algorithms in C**, including:

1. **Factorial**
2. **Fibonacci**
3. **Sum of Array Elements**
4. **Binary Search (Recursive)**

**🔸 1. Factorial using Recursion**

c

CopyEdit

#include <stdio.h>

int factorial(int n) {

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

return 1;

else

return n \* factorial(n - 1);

}

int main() {

int num = 5;

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

return 0;

}

**🔸 2. Fibonacci Series (nth term)**

c

CopyEdit

#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 n = 7;

printf("Fibonacci term at position %d is %d\n", n, fibonacci(n));

return 0;

}

**🔸 3. Sum of Array Elements (Recursive)**

c

CopyEdit

#include <stdio.h>

int sumArray(int arr[], int size) {

if (size == 0)

return 0;

return arr[size - 1] + sumArray(arr, size - 1);

}

int main() {

int arr[] = {1, 2, 3, 4, 5};

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

printf("Sum of array elements is %d\n", sumArray(arr, size));

return 0;

}

**🔸 4. Binary Search (Recursive)**

c

CopyEdit

#include <stdio.h>

int binarySearch(int arr[], int left, int right, int key) {

if (left > right)

return -1;

int mid = (left + right) / 2;

if (arr[mid] == key)

return mid;

else if (key < arr[mid])

return binarySearch(arr, left, mid - 1, key);

else

return binarySearch(arr, mid + 1, right, key);

}

int main() {

int arr[] = {2, 4, 6, 8, 10, 12, 14};

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

int key = 10;

int result = binarySearch(arr, 0, size - 1, key);

if (result != -1)

printf("Element found at index %d\n", result);

else

printf("Element not found\n");

return 0;

}

**non-recursive (iterative)** versions of common algorithms using the **C programming language**.

Here are examples of:

1. **Factorial (Iterative)**
2. **Fibonacci (Iterative)**
3. **Sum of Array Elements**
4. **Binary Search (Iterative)**

**🔸 1. Factorial (Non-Recursive)**

c

CopyEdit

#include <stdio.h>

int factorial(int n) {

int result = 1;

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

result \*= i;

}

return result;

}

int main() {

int num = 5;

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

return 0;

}

**🔸 2. Fibonacci (Non-Recursive)**

c

CopyEdit

#include <stdio.h>

int fibonacci(int n) {

int a = 0, b = 1, c;

if (n == 0) return a;

for (int i = 2; i <= n; i++) {

c = a + b;

a = b;

b = c;

}

return b;

}

int main() {

int n = 7;

printf("Fibonacci term at position %d is %d\n", n, fibonacci(n));

return 0;

}

**🔸 3. Sum of Array Elements (Iterative)**

c

CopyEdit

#include <stdio.h>

int sumArray(int arr[], int size) {

int sum = 0;

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

sum += arr[i];

}

return sum;

}

int main() {

int arr[] = {1, 2, 3, 4, 5};

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

printf("Sum of array elements is %d\n", sumArray(arr, size));

return 0;

}

**🔸 4. Binary Search (Iterative)**

c

CopyEdit

#include <stdio.h>

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

int left = 0, right = size - 1;

while (left <= right) {

int mid = (left + right) / 2;

if (arr[mid] == key)

return mid;

else if (arr[mid] < key)

left = mid + 1;

else

right = mid - 1;

}

return -1; // Not found

}

int main() {

int arr[] = {2, 4, 6, 8, 10, 12};

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

int key = 10;

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

if (result != -1)

printf("Element found at index %d\n", result);

else

printf("Element not found\n");

return 0;

}