Sorting Algorithm’s

```c
/*

Que : Write a C program for selection Sort
Owner: Rushikesh Sanjay Pokharkar
Batch: PPA9

*/

//                                        *********  Solution  *********


#include<stdio.h> //Include Necessary Header Files.

void selectionSort(int arr[], int n)
{
	// Logic to sort array in ascending order Using Selection Sort.
	for (int i = 0; i < n - 1; i++)
	{
		for (int j = i + 1; j < n; j++)
		{
			if (arr[i] > arr[j]) {
				int temp = arr[i];
				arr[i] = arr[j];
				arr[j] = temp;
			}
		}
	}
}

void main() {

	int arr[100], n, min; // Declaration of required varibales.

	printf("How many Elements do you want in array?\n");
	scanf_s("%d", &n); // Take input - Number of array elements.

	printf("Enter Array Elements: \n");
	for (int i = 0; i < n; i++) // For loop to take input array elements.
	{
		scanf_s("%d", &arr[i]);
	}

	printf("Unsorted Array Elements are: ");

	for (int i = 0; i < n; i++) // For loop to print array elements.
	{
		printf("%d ", arr[i]);
	}

	printf("\nSorted Array In Ascending Order Using Selection Sort: ");

	selectionSort(arr, n); // Function call for selection sort.

	for (int i = 0; i < n; i++) // For loop to print sorted array..
	{
		printf("%d ", arr[i]);
	}
	printf("\n");
}




/*

Que : Write a C program for Bubble Sort
Owner: Rushikesh Sanjay Pokharkar
Batch: PPA9

*/

//                                          *********  Solution  *********


#include<stdio.h> //Include Necessary Header Files.

void bubbleSort(int arr[], int n)
{
	// Logic to sort array in ascending order Using Bubble Sort.
	for (int i = 1; i < n; i++)
	{
		for (int j = 0; j < n - i; j++)
		{
			if (arr[j] > arr[j + 1])
			{
				int temp = arr[j];
				arr[j] = arr[j + 1];
				arr[j + 1] = temp;
			}
		}
	}
}

void main() {

	int arr[100], n; // Declaration of required varibales.

	printf("How many Elements do you want in array?\n");
	scanf_s("%d", &n); // Take input - Number of array elements.

	printf("Enter Array Elements: \n");
	for (int i = 0; i < n; i++) // For loop to take input array elements.
	{
		scanf_s("%d", &arr[i]);
	}

	printf("Unsorted Array Elements are: ");

	for (int i = 0; i < n; i++) // For loop to print array elements.
	{
		printf("%d ", arr[i]);
	}

	printf("\nSorted Array In Ascending Order Using Bubble Sort: ");

	bubbleSort(arr, n); // function call for bubble sort
	for (int i = 0; i < n; i++) // For loop to print sorted array..
	{
		printf("%d ", arr[i]);
	}
	printf("\n");
}




/*

Que : Write a C program for Insertion Sort
Owner: Rushikesh Sanjay Pokharkar
Batch: PPA9

*/

//                                        *********  Solution  *********


#include<stdio.h> //Include Necessary Header Files.

void insertionSort(int arr[], int n)
{
	// Logic to sort array in ascending order Using Insertion Sort.
	for (int i = 1; i < n; i++)
	{
		int temp = arr[i];
		int empty = i;

		while (empty > 0 && arr[empty - 1] > temp)
		{
			arr[empty] = arr[empty - 1];
			empty--;
		}
		arr[empty] = temp;
	}
}

void main() {

	int arr[100], n, min; // Declaration of required varibales.

	printf("How many Elements do you want in array?\n");
	scanf_s("%d", &n); // Take input - Number of array elements.

	printf("Enter Array Elements: \n");
	for (int i = 0; i < n; i++) // For loop to take input array elements.
	{
		scanf_s("%d", &arr[i]);
	}

	printf("Unsorted Array Elements are: ");

	for (int i = 0; i < n; i++) // For loop to print array elements.
	{
		printf("%d ", arr[i]);
	}

	printf("\nSorted Array In Ascending Order Using Insertion Sort: ");
	insertionSort(arr, n); // Function call for Insertion sort.

	for (int i = 0; i < n; i++) // For loop to print sorted array..
	{
		printf("%d ", arr[i]);
	}
	printf("\n");
}





/*

Que : Write a C program for Quick Sort
Owner: Rushikesh Sanjay Pokharkar
Batch: PPA9

*/

//                                          *********  Solution  *********


#include<stdio.h> //Include Necessary Header Files.

void quickSort(int arr[], int L, int H)
{
	// Logic to sort array in ascending order Using Quick Sort.
	int low = L + 1;
	int high = H;
	int pivot = arr[L];

	while (low <= high)
	{
		while (arr[low] < pivot)
		{
			low++;
		}
		while (arr[high] > pivot)
		{
			high--;
		}
		if (low <= high)
		{
			int temp = arr[low];
			arr[low] = arr[high];
			arr[high] = temp;
			low++;
			high--;
		}
	}
	int temp = arr[L];
	arr[L] = arr[high];
	arr[high] = temp;

	if (L<high && L != high-1)
	{
		quickSort(arr, L, high - 1);
	}
	if (low<H && low != H)
	{
		quickSort(arr, low, H);
	}
}

void main() {

	int arr[100], n; // Declaration of required varibales.

	printf("How many Elements do you want in array?\n");
	scanf_s("%d", &n); // Take input - Number of array elements.

	printf("Enter Array Elements: \n");
	for (int i = 0; i < n; i++) // For loop to take input array elements.
	{
		scanf_s("%d", &arr[i]);
	}

	printf("Unsorted Array Elements are: ");

	for (int i = 0; i < n; i++) // For loop to print array elements.
	{
		printf("%d ", arr[i]);
	}

	printf("\nSorted Array In Ascending Order Using Quick Sort: ");

	quickSort(arr, 0, n-1); // Function call for Insertion sort.

	for (int i = 0; i < n; i++) // For loop to print sorted array..
	{
		printf("%d ", arr[i]);
	}
	printf("\n");
}




/*

Que : Write a C program for Merge Sort
Owner: Rushikesh Sanjay Pokharkar
Batch: PPA9

*/

//                                        *********  Solution  *********


#include<stdio.h> //Include Necessary Header Files.


void merge(int arr[], int low, int mid, int high) // Function to sort the array.
{
	int temp_arr[100] ;
	int i = low, j = mid + 1, k = 0;

	while (i <= mid && j <= high)
	{
		if (arr[i] < arr[j])
		{
			temp_arr[k] = arr[i];
			i++, k++;
		}
		else
		{
			temp_arr[k] = arr[j];
			j++, k++;
		}
	}
	while (i <= mid)
	{
		temp_arr[k] = arr[i];
		i++, k++;
	}while (j <= high)
	{
		temp_arr[k] = arr[j];
		j++, k++;
	}

	for (int i = low, j = 0; i <= high; i++,j++)
	{
		arr[i] = temp_arr[j];
	}
}

void mergeSort(int arr[], int low, int high)
{
	// Logic to sort array in ascending order Using Merge Sort.
	if (low < high)
	{
		int mid = (low + high) / 2;
		if (low != mid)
		{
			mergeSort(arr, low, mid); // function call to divide first half
of array.
		}
		if (mid + 1 != high)
		{
			mergeSort(arr, mid + 1, high); // Function call to divide second
half of array.
		}
		merge(arr, low, mid, high); // Function call to merge to sorted halfs of
the array.
	}
}

void main() {

	int arr[100], n; // Declaration of required varibales.

	printf("How many Elements do you want in array?\n");
	scanf_s("%d", &n); // Take input - Number of array elements.

	printf("Enter Array Elements: \n");
	for (int i = 0; i < n; i++) // For loop to take input array elements.
	{
		scanf_s("%d", &arr[i]);
	}
```

```c
	printf("Unsorted Array Elements are: ");

	for (int i = 0; i < n; i++) // For loop to print array elements.
	{
		printf("%d ", arr[i]);
	}

	printf("\nSorted Array In Ascending Order Using Merge Sort: ");

	mergeSort(arr, 0, n - 1); // Function call for Merge sort.

	for (int i = 0; i < n; i++) // For loop to print sorted array..
	{
		printf("%d ", arr[i]);
	}
	printf("\n");
}
```

Searching Algorithm’s

```c
/*

Que : Write a C program for Linear Search
Owner: Rushikesh Sanjay Pokharkar
Batch: PPA9

*/

//                                        *********  Solution  *********


#include<stdio.h> //Include Necessary Header Files.


int linearSearch(int arr[], int n, int num)
{
	// Logic of linear search
	for (int i = 0; i < n; i++)
	{
		if (arr[i] == num)
		{
			return 1;
		}
	}
	return 0;
}

void main() {
	int arr[100], n, num; // Declaration of required varibales.

	printf("How many Elements do you want in array?\n");
	scanf_s("%d", &n); // Take input - Number of array elements.

	printf("Enter Array Elements: \n");
	for (int i = 0; i < n; i++) // For loop to take input array elements.
	{
		scanf_s("%d", &arr[i]);
	}

	printf("Array Elements are: ");

	for (int i = 0; i < n; i++) // For loop to print array elements.
	{
		printf("%d ", arr[i]);
	}
	printf("\n");

	printf("Enter a number to search in given array: ");
	scanf_s("%d", &num);

	int result = linearSearch(arr, n, num); // Function call for Linear Search

	if (result)
	{
		printf("Given element %d is present in the array.\n", num);
	}
	else
	{
		printf("Given element %d is not present in the array.\n", num);
	}

}





/*

Que : Write a C program for Two Pointer Method for searching of element in array.
Owner: Rushikesh Sanjay Pokharkar
Batch: PPA9

*/

//					*********  Solution  *********


#include<stdio.h> //Include Necessary Header Files.


int twoPointerMethod(int arr[], int n, int num)
{
	// Logic of Two Pointer Method
	int low = 0, high = n-1;
	while (low <= high)
	{
		if (arr[low] == num || arr[high] == num)
		{
				return 1;
			}
			low++, high--;
		}
		return 0;
}

void main() {

		int arr[100], n, num; // Declaration of required varibales.

		printf("How many Elements do you want in array?\n");
		scanf_s("%d", &n); // Take input - Number of array elements.

		printf("Enter Array Elements: \n");
		for (int i = 0; i < n; i++) // For loop to take input array elements.
		{
			scanf_s("%d", &arr[i]);
		}

		printf("Array Elements are: ");

		for (int i = 0; i < n; i++) // For loop to print array elements.
		{
			printf("%d ", arr[i]);
		}
		printf("\n");

		printf("Enter a number to search in given array: ");
		scanf_s("%d", &num);

		int result = twoPointerMethod(arr, n, num); // Function call for Two Pointer
Method

		if (result)
		{
			printf("Given element %d is present in the array.\n", num);
		}
		else
		{
			printf("Given element %d is not present in the array.\n", num);
		}

}





/*

Que : Write a C program for Binary Search Using Recursion
Owner: Rushikesh Sanjay Pokharkar
Batch: PPA9

*/

//                                          *********  Solution  *********


#include<stdio.h> //Include Necessary Header Files.
```

```
void merge(int arr[], int low, int mid, int high) // Function to sort the array.
{
	int temp_arr[100];
	int i = low, j = mid + 1, k = 0;

	while (i <= mid && j <= high)
	{
		if (arr[i] < arr[j])
		{
			temp_arr[k] = arr[i];
			i++, k++;
		}
		else
		{
			temp_arr[k] = arr[j];
			j++, k++;
		}
	}
	while (i <= mid)
	{
		temp_arr[k] = arr[i];
		i++, k++;
	}while (j <= high)
	{
		temp_arr[k] = arr[j];
		j++, k++;
	}

	for (int i = low, j = 0; i <= high; i++, j++)
	{
		arr[i] = temp_arr[j];
	}
}

void mergeSort(int arr[], int low, int high)
{
	// Logic to sort array in ascending order Using Merge Sort.
	if (low < high)
	{
		int mid = (low + high) / 2;
		if (low != mid)
		{
			mergeSort(arr, low, mid); // function call to divide first half
of array.
		}
		if (mid + 1 != high)
		{
			mergeSort(arr, mid + 1, high); // Function call to divide second
half of array.
		}
		merge(arr, low, mid, high); // Function call to merge to sorted halfs of
the array.
	}
}


int binarySearch(int arr[], int low, int high, int num)
{
	// Logic of binary search Using recursion.
	if (num >= arr[low] && num <= arr[high])
```

```c
	{
		int mid = (low + high) / 2;
		if (arr[mid] == num)
		{
			return 1;
		}
		else if (num < arr[mid])
		{
			return binarySearch(arr, low, mid - 1, num);
		}
		else
		{
			return binarySearch(arr, mid + 1, high, num);
		}
	}
	return 0;
}

void main() {

	int arr[100], n, num; // Declaration of required varibales.

	printf("How many Elements do you want in array?\n");
	scanf_s("%d", &n); // Take input - Number of array elements.

	printf("Enter Array Elements: \n");
	for (int i = 0; i < n; i++) // For loop to take input array elements.
	{
		scanf_s("%d", &arr[i]);
	}

	printf("Array Elements are: ");

	for (int i = 0; i < n; i++) // For loop to print array elements.
	{
		printf("%d ", arr[i]);
	}
	printf("\n");


	mergeSort(arr, 0, n - 1); // Function call to sort the array.

	printf("Sorted Array Elements are: ");

	for (int i = 0; i < n; i++) // For loop to print array elements.
	{
		printf("%d ", arr[i]);
	}
	printf("\n");


	printf("Enter a number to search in given array: ");
	scanf_s("%d", &num);

	int result = binarySearch(arr, 0, n - 1, num); // Function call for binary 
Search using recursion.

	if (result)
	{
		printf("Given element %d is present in the array.\n", num);
	}
	else
	{
		printf("Given element %d is not present in the array.\n", num);
	}

}





/*

Que : Write a C program for Binary Search Without Using Recursion
Owner: Rushikesh Sanjay Pokharkar
Batch: PPA9

*/

//                                          *********  Solution  *********


#include<stdio.h> //Include Necessary Header Files.


void merge(int arr[], int low, int mid, int high) // Function to sort the array.
{
	int temp_arr[100];
	int i = low, j = mid + 1, k = 0;

	while (i <= mid && j <= high)
	{
		if (arr[i] < arr[j])
		{
			temp_arr[k] = arr[i];
			i++, k++;
		}
		else
		{
			temp_arr[k] = arr[j];
			j++, k++;
		}
	}
	while (i <= mid)
	{
		temp_arr[k] = arr[i];
		i++, k++;
	}while (j <= high)
	{
		temp_arr[k] = arr[j];
		j++, k++;
	}

	for (int i = low, j = 0; i <= high; i++, j++)
	{
		arr[i] = temp_arr[j];
	}
}

void mergeSort(int arr[], int low, int high)
{
	// Logic to sort array in ascending order Using Merge Sort.
	if (low < high)
	{
		int mid = (low + high) / 2;
		if (low != mid)
		{
			mergeSort(arr, low, mid); // function call to divide first half 
of array.
		}
		if (mid + 1 != high)
		{
			mergeSort(arr, mid + 1, high); // Function call to divide second 
half of array.
		}
		merge(arr, low, mid, high); // Function call to merge to sorted halfs of 
the array.
	}
}


int binarySearch(int arr[], int n, int num)
{
	// Logic of binary search without Using recursion.
	int low = 0;
	int high = n - 1;

	while ((num >= arr[low] && num <= arr[high]) || low <= high)
	{
		int mid = (low + high) / 2;
		if (arr[mid] == num)
		{
			return 1;
		}
		else if (num < arr[mid])
		{
			high = mid - 1;
		}
		else
		{
			low = mid + 1;
		}
	}
	return 0;
}

void main() {

	int arr[100], n, num; // Declaration of required varibales.

	printf("How many Elements do you want in array?\n");
	scanf_s("%d", &n); // Take input - Number of array elements.

	printf("Enter Array Elements: \n");
	for (int i = 0; i < n; i++) // For loop to take input array elements.
	{
		scanf_s("%d", &arr[i]);
	}

	printf("Array Elements are: ");

	for (int i = 0; i < n; i++) // For loop to print array elements.
	{
		printf("%d ", arr[i]);
	}
	printf("\n");


	mergeSort(arr, 0, n - 1); // Function call to sort the array.

	printf("Sorted Array Elements are: ");

	for (int i = 0; i < n; i++) // For loop to print array elements.
	{
		printf("%d ", arr[i]);
	}
	printf("\n");


	printf("Enter a number to search in given array: ");
	scanf_s("%d", &num);

	int result = binarySearch(arr, n, num); // Function call for binary Search 
without using recursion.

	if (result)
	{
		printf("Given element %d is present in the array.\n", num);
	}
	else
	{
		printf("Given element %d is not present in the array.\n", num);
	}

}
```

Stack

```
/*

Que : Write a C program for Creation of Stack.
Owner: Rushikesh Sanjay Pokharkar
Batch: PPA9

*/

//                                          *********  Solution  *********


#include<stdio.h> //Include Necessary Header Files.
```

```c
#define MAX 5

struct STACK
{
	int arr[MAX];
	int top;
};

int isFull(struct STACK* stackptr)
{
	if (stackptr->top == MAX - 1)
	{
		return 1;
	}
	return 0;
}

int isEmpty(struct STACK* stackptr)
{
	return (stackptr->top == -1);
}

void initStack(struct STACK* stackptr)
{
	stackptr->top = -1;
}

void push(struct STACK* stackptr, int data)
{
	(stackptr->top)++;
	stackptr->arr[stackptr->top] = data;
}

int pop(struct STACK* stackptr)
{
	int num = stackptr->arr[stackptr->top];
	(stackptr->top)--;
	return num;
}

void main()
{
	int choice;
	struct STACK stack;
	initStack(&stack);

	do
	{
		printf("Enter Your Choice: \n");
		printf("0. Exit.\n");
		printf("1. Push.\n");
		printf("2. Pop.\n");
		//printf("3. Display.\n");
		printf("Choice = ");
		scanf_s("%d", &choice);

		switch (choice)
		{
		case 0:
			printf("Thank You!!!\n");
			break;
		case 1:
```

```
                    if (isFull(&stack))
                    {
                          printf("Stack is FULL You can not perform PUSH Operation
on it.\n");
                    }
                    else
                    {
                          int data;
                          printf("Enter a data: ");
                          scanf_s("%d", &data);
                          push(&stack, data);
                    }
                    break;
              case 2:
                    if (isEmpty(&stack))
                    {
                          printf("Stack is Empty You can not perform POP Operation
on it.\n");
                    }
                    else
                    {
                          printf("The value %d is Poped From Stack.\n",
pop(&stack));
                    }
                    break;
              default:
                    printf("Please Enter a Valid Choice.\n");
              }
       } while (choice != 0);
}
```