Largest & Smallest number in an Array

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

void main(){
	int arr[10], i, large, small;
	printf("Enter 10 Numbers\n");
	for (i = 0; i < 10; i++){
		scanf("%d", &arr[i]);
	}
	large = arr[0];		small = arr[0];
	for (i = 0; i < 10; i++){
		if (arr[i] > large){
			large = arr[i];
		}
		if(arr[i] < small){
			small = arr[i];
		}
	}
	printf("Largest number is %d\n", large);
	printf("Smallest number is %d", small);
	getch();
}
```

2-D Array (3x4)

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

void main(){
	int arr[3][4], i, j;
	printf("Enter 12 numbers:\n");
	for(i = 0; i < 3; i++){
		for(j = 0; j < 4; j++){
			scanf("%d", &arr[i][j]);
		}
	}
	printf("\nThe Array is:\n");
	for(i = 0; i < 3; i++){
		for(j = 0; j < 4; j++){
			printf("%d ", arr[i][j]);
		}
		printf("\n");
	}
	getch();
}
```

Sum of Array

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

void main() {
	int i, arr[10], sum=0;
	printf("Enter 10 numbers ");
	for (i = 0; i < 10; i++){
		scanf("%d", &arr[i]);
	}
	for (i = 0; i < 10; i++){
		sum = sum + arr[i];
	}
	printf("The Sum of Array is %d", sum);
	getch();
}
```

Structure

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

struct student{
	int roll_no;
	char name [16];
	float percent;
};
void main(){
	struct student s1, s2;
	printf("Enter Roll no, Name & Percentage of Student 1: ");
	scanf("%d %s %f", &s1.roll_no, s1.name, &s1.percent);
	printf("Enter Roll no, Name & Percentage of Student 2: ");
	scanf("%d %s %f", &s2.roll_no, s2.name, &s2.percent);
	getch();
}
```

Array of Structure

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

struct student{
	int roll_no;
	char name[16];
	float percent;
};
void main(){
	struct student s[5];
	int i;
	for (i = 1; i <= 2; i++){
		printf("Enter information regarding Student %d:\n", i);
		scanf("%d%s%f", &s[i].roll_no, s[i].name, &s[i].percent);
	}	printf("\n");
	for (i = 1; i <= 2; i++){
		printf("Roll No = %d\t Name = %s\t Percentage = %f\n", s[i].roll_no, s[i].name, s[i].percent);
	}
	getch();
}
```

Linear Search

```
#include <stdio.h>
#include <conio.h>
void main(){
	int arr[50], i, n, val, pos, flag = 0;
	printf("Enter the no. of elements in Array: ");
	scanf("%d", &n);
	printf("Enter %d elements in Array:\n", n);
	for (i = 0; i < n; i++){
		scanf("%d", &arr[i]);
	}
	printf("Enter the element to be found: ");
	scanf("%d", &val);
	for (i = 0; i < n; i++){
		if (arr[i] == val){
			flag = 1;		pos = i;		break;
		}
	}
	if (flag == 0)
		printf("Element not found!");
	else
		printf("The element is at %d position.", pos+1);
	getch();
}
```

Binary Search

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

void main(){
	int arr[50], i, n, high, low, val, mid, pos, flag=0;
	printf("Enter the no. of elements in Array: ");
	scanf("%d", &n);
	printf("Enter %d elements in Array in ascending order:\n", n);
	for (i = 0; i < n; i++){
		scanf("%d", &arr[i]);
	}
	printf("Enter the element to be found: ");
	scanf("%d", &val);
	high = n-1;	low = 0;
	while(low <= high){
		mid = (low + high)/2;
		if(arr[mid] == val){
			pos = mid;	flag = 1;
			break;
		}
		else if (val > arr[mid])
			low = mid + 1;
		else
			high = mid - 1;
	}
	if (flag == 1)
		printf("The element is at %d position.", pos+1);
	else
		printf("Element Not Found!");
	getch();
}
```

Bubble Sort

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

void main(){
	int arr[20], i, j, n, tmp;
	printf("Enter the No. of Elements: ");
	scanf("%d", &n);
	printf("Enter %d elements: ", n);
	for(i = 0; i < n; i++){
		scanf("%d", &arr[i]);
	}
	for(i = 0; i < n-1; i++){
		for(j = 0; j < n-1-i; j++){
			if (arr[j] > arr[j+1]){
				tmp = arr[j];	arr[j] = arr[j+1];
				arr[j+1] = tmp;
			}
		}
	}
	printf("Sorted array is: ");
	for(i = 0; i < n; i++){
		printf("%d ", arr[i]);
	}
	getch();
}
```

Selection Sort

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

void main(){
	int arr[20], i, j, n, tmp, min, pos;
	printf("Enter the No. of Elements: ");
	scanf("%d", &n);
	printf("Enter %d elements: ", n);
	for(i = 0; i < n; i++){
		scanf("%d", &arr[i]);
	}
	for(i = 0; i < n-1; i++){
		min = arr[i];
		pos = i;
		for(j = i+1; j <= n-1; j++){
			if (min > arr[j]){
				min = arr[j];	pos = j;
			}
		}
		tmp = arr[pos];		arr[pos] = arr[i];		arr[i] = tmp;
	}
	printf("Sorted array is: ");
	for(i = 0; i < n; i++){
		printf("%d ", arr[i]);
	}
	getch();
}
```

Insertion Sort (11-9-19)

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

void main(){
	int arr[20], i, j, n, key;
	printf("Enter the No. of Elements: ");
	scanf("%d", &n);
	printf("Enter %d elements: ", n);
	for(i = 0; i < n; i++){
		scanf("%d", &arr[i]);
	}
	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;
	}
	printf("Sorted array is: ");
	for(i = 0; i < n; i++){
		printf("%d ", arr[i]);
	}
	getch();
}
```

Pointers

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

void main(){
	int *ptr1, a, b;
	float *ptr2, c;
	printf("Enter A: ");		scanf("%d", &a);
	ptr1 = &a; //address of a
	printf("Value of A = %d, %d, %d\n", a, *ptr1, *(&a));
	printf("Address of A = %d, %d\n\n", &a, ptr1);
	printf("Enter B: ");		scanf("%d", &b);
	ptr1 = &b;
	printf("Value of B = %d, %d\n\n", b, *ptr1);
	printf("Enter C: ");		scanf("%f", &c);
	ptr2 = &c;
	printf("Value of C = %f, %f", c, *ptr2);
	getch();
}
```

Array using Pointers

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

void main(){
	int *p, a[5], i;
	p = a; //base address of array
	printf("Enter 5 elements in Array:\n");
	for(i = 0; i < 5; i++){
		scanf("%d", p+i);
	}
	printf("Elements of Array:\n");
	for(i = 0; i < 5; i++){
		printf("%d ", *(p+i));
	}
	getch();
}
```

Linked Lists

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>

void insert();
void display();
void insert_at_start();
void insert_after_value();
void sort_list();
void search();
void reverse();
void delete_first_node();
void delete_last_node();
void delete_node();

struct node{
	int info;
	struct node *link;
};
typedef struct node node; //rename struct node to node
node *start=NULL, *ptr, *nw;

void insert(){
	node *ptr, *nw;
	int val;
	printf("Enter value to be inserted: ");
	scanf("%d", &val);
	fflush(stdin);
	ptr = (node *)malloc(sizeof(node)); // * -- datatype
	ptr -> info = val;		ptr -> link = NULL;
	if(start == NULL){
		start = ptr;		ptr = NULL;
	}
	else{
		nw = start;
		while (nw -> link != NULL){
			nw = nw -> link;
		}
		nw -> link = ptr;
	}
}

void insert_at_start(){
	node *ptr;
	int val;
	ptr = (node *)malloc(sizeof(node));
	printf("Enter value to insert: ");
	scanf("%d", &val);
	ptr -> info = val;		ptr -> link = NULL;
	ptr -> link = start;
	start = ptr;
	ptr = NULL;
}
```

```
void insert_after_value(){
	node *ptr;
	int val1, val2, flag = 0;
	printf("Enter value to insert: ");
	scanf("%d", &val1);
	nw = (node *)malloc(sizeof(node));
	nw -> info = val1;		nw -> link = NULL;
	printf("Enter value after which to insert new value: ");
	scanf("%d", &val2);
	ptr = start;
	while (ptr != NULL){
		if (ptr -> info == val2){
			flag = 1;		break;
		}
		else
			ptr = ptr -> link;
	}
	if (flag == 0) {
		printf("Value Cannot be Inserted");
		exit(1);
	}
	else {
		nw -> link = ptr -> link;
		ptr -> link = nw;
	}
}

void sort_list(){
	node *i, *j;
	int temp;
	for (i = start; i -> link != NULL; i = i -> link){
		for (j = i -> link; j != NULL; j = j -> link){
			if (i -> info > j -> info){
				temp = i -> info;
				i -> info = j -> info;
				j -> info = temp;
			}
		}
	}
}

void reverse(){
	node *prev, *next, *current;
	prev = NULL;
	next = NULL;
	current  = start;
	while(current != NULL){
		next = current -> link;
		current -> link = prev;
		prev = current;
		current = next;
	}
	start = prev;
}
```

```
void search(){
	node *ptr;
	int val, flag=0, count=0;
	printf("Enter the value to search: ");
	scanf("%d", &val);
	ptr = start;
	while (ptr != NULL){
		count++;
		if(ptr -> info == val){
			flag = 1;		break;
		}
		else
			ptr = ptr -> link;
	}
	if(flag == 0)
		printf("Value Not Present");
	else{
		printf("Position of Node = %d\n", count);
		printf("Address of Node = %d\n", ptr); //ptr -> info (to print value)
	}
}

void delete_first_node(){
	start = start -> link;
}

void delete_node(){
	node *ptr1, *ptr2;
	int val, flag = 0;
	printf("Enter value to delete: ", val);
	scanf("%d", &val);
	ptr1 = NULL;
	ptr2 = start;
	while(ptr2 != NULL){
		if (ptr2 -> info == val){
			flag = 1;		break;
		}
		ptr1 = ptr2;		ptr2 = ptr2 -> link;
	}
	ptr1 -> link = ptr2 -> link;
	ptr1 = NULL;	ptr2 = NULL;
}

void delete_last_node(){
	node *ptr1, *ptr2;
	if (start == NULL)
		printf("The Linked List is Empty!\n");
	else if (start -> link == NULL){
		start = NULL;		printf("Last remaining node deleted.\n");
	}
	else{
		ptr1 = start;
		while (ptr1 -> link != NULL){
			ptr2 = ptr1;		ptr1 = ptr1 -> link;
		}
		ptr2 -> link = NULL;		printf("Last node deleted.\n");
	}
}
```

```
void display(){
	node *temp;
	temp = start;
	printf("The Linked List is: ");
	while (temp != NULL){
		printf("%d ", temp -> info);
		temp = temp -> link;
	}
	printf("\n");
}

void main(){
	int choice;
	printf("Enter 1 to Create Linked List.\n");
	printf("Enter 2 to Insert new value at start.\n");
	printf("Enter 3 to Insert value after specific value.\n");
	printf("Enter 4 to Sort the Linked List.\n");
	printf("Enter 5 to Search the Linked List.\n");
	printf("Enter 6 to Reverse the Linked List.\n");
	printf("Enter 7 to Delete First value.\n");
	printf("Enter 8 to Delete specific value.\n");
	printf("Enter 9 to Delete Last value.\n");
	printf("Enter 0 to Display Linked List.\n");
	printf("Enter 11 to Exit.\n");
	while(1){
		printf("\nEnter your choice: ");
		scanf("%d", &choice);
		switch(choice){
			case 1: insert();			break;
			case 2: insert_at_start();		break;
			case 3: insert_after_value();	break;
			case 4: sort_list();			break;
			case 5: search();			break;
			case 6: reverse();			break;
			case 7: delete_first_node();	break;
			case 8: delete_node();		break;
			case 9: delete_last_node();	break;
			case 0: display();			break;
			case 11: exit(1);
				break;
			default: printf("Invalid Choice!\n");
		}
	}
	getch();
}
```

Stack using Array

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define size 10

void PUSH();
void POP();
void display();
int top = -1, stack[size], val;

void PUSH(){
	if(top == size-1)
		printf("Stack is Full. Cannot PUSH");
	else{
		printf("Enter value to PUSH: ");
		scanf("%d", &val);
		top = top + 1;		stack[top] = val;
	}
}

void POP(){
	if(top == -1)
		printf("Stack is Empty. Cannot POP");
	else{
		val = stack[top];top = top - 1;
		printf("Popped element is %d\n", val);
	}
}

void display(){
	int i;
	for (i = top; i >= 0; i--){
		printf("%d ", stack[i]);
	}
	printf("\n");
}

void main(){
	int choice;
	printf("Enter 1 to PUSH.\n");
	printf("Enter 2 to POP.\n");
	printf("Enter 3 to display stack.\n");
	printf("Enter 4 to exit.\n");
	while(1){
		printf("\nEnter your choice: ");
		scanf("%d", &choice);
		switch(choice){
			case 1: PUSH();break;
			case 2: POP();		break;
			case 3: display();	break;
			case 4: exit(1);
			default: printf("Invalid Value!\n");
		}
	}
	getch();
}
```

Stack using Linked List

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>

void PUSH();
void POP();
void display();
struct node{
	int info;
	struct node *link;
};
typedef struct node node;
int val;
node *top = NULL, *ptr;

void PUSH(){
	printf("Enter value to PUSH: ");
	scanf("%d", &val);
	ptr = (node *)malloc(sizeof(node));
	ptr -> info = val; ptr -> link = NULL;
	if (top == NULL){
		top = ptr;		ptr = NULL;
	}
	else{
		ptr -> link = top; top = ptr;		ptr = NULL;
	}
}
void POP(){
	if(top == NULL)
		printf("Stack is Empty. Cannot POP");
	else{
		top = top -> link;
		printf("Popped element is %d\n", val);
	}
}
void display(){
	ptr = top;
	while(ptr != NULL){
		printf("%d ", ptr -> info);
		ptr = ptr -> link;
	}
	printf("\n");
}

void main(){
	int choice;
	printf("Enter 1 to PUSH.\n");
	printf("Enter 2 to POP.\n");
	printf("Enter 3 to display.\n");
	printf("Enter 0 to exit.\n");
	while(1){
		printf("\nEnter your choice: ");
		scanf("%d", &choice);
```

```
		switch(choice){
			case 1: PUSH(); break;
			case 2: POP();		break;
			case 3: display();	break;
			case 0: exit(1);
			default: printf("Invalid Value!\n");
		}
	}
	getch();
}
```

Queue using Array

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define size 10

void insert();
void del();
void display();
int front = -1, rear = -1, queue[size];

void insert(){
	int val;
	printf("Enter value to insert: ");
	scanf("%d", &val);
	if(front == 0 && rear == size - 1 || front == rear + 1)
		printf("Queue is Full - Overflow.");
	else if(rear == -1){
		rear = front = 0; queue[rear] = val;
	}
	else if(rear == size - 1){
		rear = 0;			queue[rear] = val;
	}
	else{
		rear = rear + 1;		queue[rear] = val;
	}
}

void del(){
	if (front == -1)
		printf("Queue is Empty - Underflow.");
	else if(front == rear)
		front = rear = -1;
	else if(front == size - 1)
		front = 0;
	else
		front = front + 1;
}
```

```
void display(){
	int i;
	if (front == -1)
		printf("Queue is Empty - Underflow.");
	if(front <= rear){
		for (i = front; i <= rear; i++){
			printf("%d ", queue[i]);
		}
	}
	else{
		for(i = front; i <= rear; i++){
			printf("%d ", queue[i]);
		}
		for(i = 0; i <= rear; i++){
			printf("%d ", queue[i]);
		}
	}
	printf("\n");
}

void main(){
	int choice;
	printf("Enter 1 to insert.\n");
	printf("Enter 2 to delete.\n");
	printf("Enter 3 to display queue.\n");
	printf("Enter 0 to exit.\n");
	while(1){
		printf("\nEnter your choice: ");
		scanf("%d", &choice);
		switch(choice){
			case 1: insert(); break;
			case 2: del();		break;
			case 3: display();	break;
			case 0: exit(1);
			default: ("Invalid Choice!");
		}
	}
	getch();
}
```

Queue using Linked List

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>

void insert();
void del();
void display();

struct node{
	int info;
	struct node *link;
};
typedef struct node node;
int val;
node *front = NULL, *rear = NULL, *ptr;
```

```
void insert(){
	printf("Enter value to insert: ");
	scanf("%d", &val);
	ptr = (node *)malloc(sizeof(node));
	ptr -> info = val;
	ptr -> link = NULL;



	if (rear == NULL){
		front = rear = ptr;		ptr = NULL;
	}
	else{
		rear -> link = ptr;		rear = ptr;		ptr = NULL;
	}
}

void del(){
	if(front == NULL)
		printf("Queue is Empty. Cannot delete");
	else
		front = front -> link;
		printf("Deleted element is %d\n", val);
}

void display(){
	ptr = front;
	while(ptr != NULL){
		printf("%d ", ptr -> info);
		ptr = ptr -> link;
	}
	printf("\n");
}

void main(){
	int choice;
	printf("Enter 1 to insert value in Queue.\n");
	printf("Enter 2 to delete value from Queue.\n");
	printf("Enter 3 to display the Queue.\n");
	printf("Enter 0 to exit.\n");
	while(1){
		printf("\nEnter your choice: ");
		scanf("%d", &choice);
		switch(choice){
			case 1: insert(); break;
			case 2: del();		break;
			case 3: display();	break;
			case 0: exit(1);
			default: printf("Invalid Value!");
		}
	}
	getch();
}
```