1.Reversing a 32-bit signed intergers:
code:

```
#include <stdio.h>
int reversDigits(int num)
{
    int rev_num = 0;
    while (num != 0) {
        rev_num = rev_num * 10 + num % 10;
        num = num / 10;
    }
    return rev_num;
}
int main()
{
    int num = 5896;
    printf("Reverse of   number   is %d", reversDigits(num));
    return 0;
}
```

Output: Reverse of number is 6985

```
Reverse of  number  is 6985

=== Code Execution Successful ===
```

2.Vaild string no not:
code:

```
#include <stdio.h>
#include <stdbool.h>
#define MAX_SIZE 100
int main() {
    char stack[MAX_SIZE];
    int top = -1;
    char mapping[MAX_SIZE];
    mapping[')'] = '(';
    mapping[']'] = '[';
    mapping['}'] = '{';
```

```
    char *strings[] = {"((()))", "([{()}])", "(()", "([)]"};
    for (int k = 0; k < sizeof(strings) / sizeof(strings[0]); k++) {
        char *s = strings[k];
        top = -1;
        bool valid = true;
        for (int i = 0; s[i] != '\0'; i++) {
            if (s[i] == '(' || s[i] == '[' || s[i] == '{') {
                stack[++top] = s[i];
            } else if (s[i] == ')' || s[i] == ']' || s[i] == '}') {
                if (top == -1 || stack[top] != mapping[s[i]]) {
                    valid = false;
                    break;
                } else {
                    top--;
                }
            }
        }
        if (top != -1) valid = false;
        printf("%s is %s\n", s, valid ? "valid" : "not valid");
    }
    return 0;
}
```

Output: ((( ))) is valid
([{( )}]) is valid
(( ) is not valid
([ )] is not valid

```
((())) is valid
([{()}]) is valid
(() is not valid
([)] is not valid


=== Code Execution Successful ===
```

3.Merging two arrays:

code:

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

```
int main(){
        int arr1size = 5, arr2size = 5, arr_resultsize, i, j;
        int a[5] = { 1, 2, 3, 4, 5 };
        int b[5] = { 6, 7, 8, 9, 10 };
        arr_resultsize = arr1size + arr2size;
        int c[arr_resultsize];
        for (i = 0; i < arr1size; i++) {
                c[i] = a[i];
        }
        for (i = 0, j = arr1size;j < arr_resultsize && i < arr2size; i++, j++) {
                c[j] = b[i];
        }
             printf("Merging of arrays is ");
        for (i = 0; i < arr_resultsize; i++) {
                printf("%d ", c[i]);
        }
        return 0;
}
```

Output: 1 2 3 4 5 6 7 8 9 10

```
Merging of arrays is 1 2 3 4 5 6 7 8 9 10

=== Code Execution Successful ===
```

4.Array Finding Duplication Values:

code:

```
#include <stdio.h>
int main()
{
    int arr[] = {1, 2, 3, 4, 2, 7, 8, 8, 3};
    int length = sizeof(arr)/sizeof(arr[0]);
    printf("Duplicate elements in given array:   ");
    for(int i = 0; i < length; i++) {
         for(int j = i + 1; j < length; j++) {
              if(arr[i] == arr[j])
                   printf("%d ", arr[j]);
         }
    }
    return 0;
}
```

output: Duplicate elements in given array : 2,3,8

```
Duplicate elements in given array: 2 3 8

=== Code Execution Successful ===
```

5.Merging of :
code:

```
#include <stdio.h>
#define MAX_SIZE 100
void merge_arrays(int arr1[], int size1, int arr2[], int size2, int merged[]) {
    int i, j, k;
    for (i = 0; i < size1; i++) {
        merged[i] = arr1[i];
    }
    for (j = 0, k = size1; j < size2; j++, k++) {
        merged[k] = arr2[j];
    }
}
int main() {
    int arr1[] = {1, 2, 3};
    int size1 = sizeof(arr1) / sizeof(arr1[0]);
    int arr2[] = {4, 5, 6};
    int size2 = sizeof(arr2) / sizeof(arr2[0]);
    int merged[MAX_SIZE];
    merge_arrays(arr1, size1, arr2, size2, merged);
    printf("Merged list: ");
    for (int i = 0; i < size1 + size2; i++) {
        printf("%d ", merged[i]);
    }
    printf("\n");
    return 0;
}
```

output: Merged list: 1 2 3 4 5 6

```
Merged list: 1 2 3 4 5 6


=== Code Execution Successful ===
```

6.Registration Number Search:

code:

```
#include <stdio.h>
#include <stdbool.h>
#define MAX_SIZE 100
bool search_registration_number(int reg_numbers[], int size, int target) {
    for (int i = 0; i < size; i++) {
        if (reg_numbers[i] == target) {
            return true;
        }
    }
    return false;
}
int main() {
    int reg_numbers[MAX_SIZE] = {123, 456, 789, 1011, 1213, 1234};
    int size = 6;
    int target = 1234;
    if (search_registration_number(reg_numbers, size, target)) {
        printf("Registration number %d found!\n", target);
    } else {
        printf("Registration number %d not found.\n", target);
    }
    return 0;
}
```

output: Registration number 1234 found!

```
Registration number 1234 found!


=== Code Execution Successful ===
```

7.Identify location of Element:
code:

```
#include <stdio.h>
#define MAX_SIZE 100
int find_element(int arr[], int size, int target) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == target) {
            return i;
        }
    }
    return -1;
}
```

```
int main() {
    int arr[MAX_SIZE] = {10, 20, 30, 40, 50};
    int size = 5;
    int target = 40;
    int element = find_element(arr, size, target);
    if (element!= -1) {
        printf("Element %d found at element %d.\n", target,element);
    } else {
        printf("Element %d not found in the array.\n", target);
    }
    return 0;
}
```

ouput: Element 40 found at element 3.

```
Element 40 found at element 3.


=== Code Execution Successful ===
```

8.Array Odd and even Values:

code:

```
#include <stdio.h>
void main()
{
    int n;
    printf("Enter number of elements in the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d elements in the array: ",n);
    for(int i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    printf("Even numbers in the array are: ");
    for(int i=0;i<n;i++)
    {
        if(arr[i]%2==0)
            printf("%d ", arr[i]);
    }
    printf("\n Odd numbers in the array are: ");
```

```
    for(int i=0;i<n;i++)
    {
        if(arr[i]%2==1)
            printf("%d ", arr[i]);
    }
}
```

Output: Enter number of elements in the array: 5
Enter 5 elements in the array: 1,2,3,4,5
Even numbers in the array are: 2 4
Odd numbers in the array are: 1 3 5

```
Enter number of elements in the array: 5
Enter 5 elements in the array: 1 2 3 4 5
Even numbers in the array are: 2 4
Odd numbers in the array are: 1 3 5

=== Code Exited With Errors ===
```

9.Sum of Fibonacci series:

code:

```
#include <stdio.h>
unsigned long long fibonacci_sum(int n) {
    if (n <= 0) {
        return 0;
    }
    long sum = 0;
    long a = 0, b = 1, temp;
    for (int i = 1; i <= n; i++) {
        sum += b;
        temp = a + b;
        a = b;
        b = temp;
    }
    return sum;
}
int main() {
    int n = 10;
    long sum = fibonacci_sum(n);
```

```
    printf("Sum of first %d Fibonacci numbers is: %lu\n", n, sum);
    return 0;
}
```

Output: Sum of first 10 Fibonacci numbers is: 143

```
Sum of first 10 Fibonacci numbers is: 143


=== Code Execution Successful ===
```

10.Factorial of a number:

code:

```
#include <stdio.h>
int main() {
    int n, i;
    long fact = 1;
    printf("Enter an integer: ");
    scanf("%d", &n);
    if (n < 0)
        printf("Error! Factorial of a negative number doesn't exist.");
    else {
        for (i = 1; i <= n; ++i) {
            fact *= i;
        }
        printf("Factorial of %d = %lu", n, fact);
    }
    return 0;
}
```

Output: Enter an integer: 5
Factorial of 5 = 120

```
Enter an integer: 5
Factorial of 5 = 120

=== Code Execution Successful ===
```