EXERCISE-26

AIM: To write a C program to reverse a 32-bit signed integer, and handle overflow cases where the reversed integer goes beyond the 32-bit signed integer range.

Algorithm:

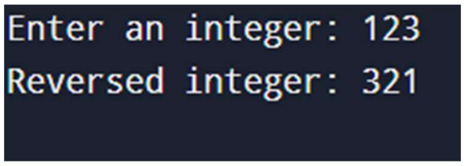
```
1. Start.
2. Read the input integer x.
3. Initialize rev = 0.
4. While x is not 0:
5. Get the last digit: pop = x % 10
6. Remove the last digit from x: x = x / 10
7. Check for overflow before appending the digit:
      a. If rev > INT_MAX/10 or rev == INT_MAX/10 and pop > 7,
         return 0 (overflow).
      b. If rev < INT_MIN/10 or rev == INT_MIN/10 and pop < -8,
         return 0 (underflow).
8. Update reversed number: rev = rev * 10 + pop
9. Print the reversed number.
10.End.
```

Program Code:

```
#include <stdio.h>

#include <limits.h>

int reverse(int x) {

   int rev = 0;
```

```
    while (x != 0) {
        int pop = x % 10;
        x /= 10;
        if (rev > INT_MAX/10 || (rev == INT_MAX / 10 && pop > 7))
return 0;
        if (rev < INT_MIN/10 || (rev == INT_MIN / 10 && pop < -8))
return 0;
        rev = rev * 10 + pop;
    }
    return rev;
}
int main() {
    int x;
    printf("Enter an integer: ");
    scanf("%d", &x);
    int result = reverse(x);
    printf("Reversed integer: %d\n", result);
    return 0;
}
```

Input and Output:



```
Enter an integer: 123
Reversed integer: 321
```

Result:

The program correctly reverses a 32-bit signed integer and handles overflow by returning 0 when the result is out of the valid range.