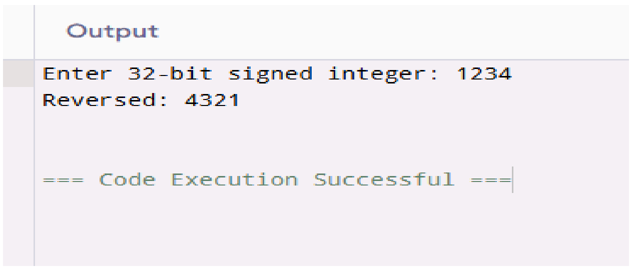
EXP NO 1: Implement a C Program for Reversing a 32 bit signed integers.

```
#include <stdio.h>

#include <limits.h>

int reverse_int(int x) {

    long long rev = 0;

    while (x != 0) {

        int digit = x % 10;

        rev = rev * 10 + digit;

        if (rev > INT_MAX || rev < INT_MIN) return 0; // overflow

        x /= 10;

    }

    return (int)rev;

}

int main() {

    int n;

    printf("Enter 32-bit signed integer: ");

    if (scanf("%d", &n) != 1) return 0;

    int r = reverse_int(n);

    if (r == 0) printf("Reversed value is 0 (possible overflow or original reversed to 0).\n");

    else printf("Reversed: %d\n", r);

    return 0;

}
```

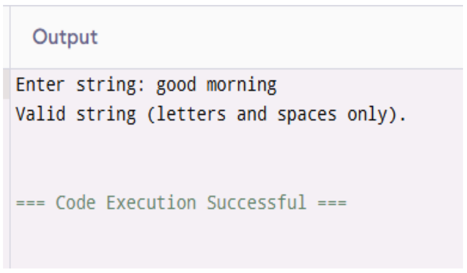

Output

```
Enter 32-bit signed integer: 1234
Reversed: 4321


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

EXP NO 2: Implement a C Program to Check for a valid String

```
#include <stdio.h>

#include <ctype.h>
```

```
#include <string.h>

int is_valid_string(const char *s) {

    for (int i = 0; s[i]; ++i) {

        if (!(isalpha((unsigned char)s[i]) || isspace((unsigned char)s[i]))) return 0;

    }

    return 1;

}

int main() {

    char buf[1024];

    printf("Enter string: ");

    if (!fgets(buf, sizeof(buf), stdin)) return 0;

    // remove newline

    buf[strcspn(buf, "\n")] = 0;

    if (is_valid_string(buf)) printf("Valid string (letters and spaces only).\n");

    else printf("Invalid string (contains non-letters/non-space characters).\n");

    return 0;

}
```

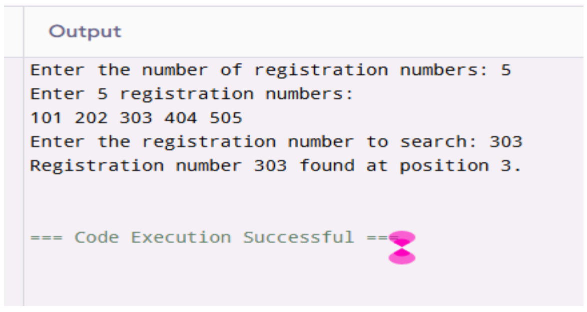


Output

```
Enter string: good morning
Valid string (letters and spaces only).


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

EXP NO 3:- earch for a particular registration number in an array[LINEAR SEARCH]

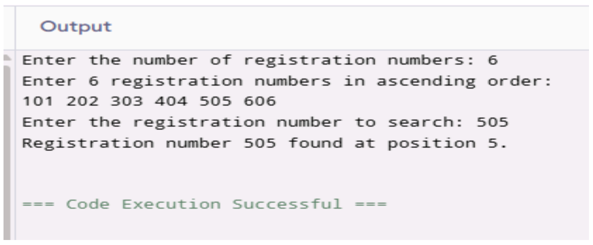
```
#include <stdio.h>

#include <string.h>

int main() {

    int n;

    printf("Enter number of registration numbers: ");

    scanf("%d", &n);

    char regs[n][100];

    for (int i = 0; i < n; i++) {
        printf("Reg no %d: ", i + 1);

        scanf("%s", regs[i]);

    }

char key[100];

    printf("Enter registration number to search: ");

    scanf("%s", key);

    int found = 0;

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

        if (strcmp(regs[i], key) == 0) {

            printf("Found at index %d (0-based)\n", i);

            found = 1;

            break;

        }

    }

if (!found)

        printf("Not found\n");

    return 0;

}
```



```
Enter the number of registration numbers: 5
Enter 5 registration numbers:
101 202 303 404 505
Enter the registration number to search: 303
Registration number 303 found at position 3.


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

EXP NO 4: Search for a particular registration number in an array[BINARY SEARCH]

```
#include <stdio.h>

#include <string.h>

int main() {

    int n;

    printf("Enter number of registration numbers: ");
    scanf(“%d“, &n);

    char regs[n][100];

    printf(“Enter registration numbers (sorted order):|n“);

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

        printf(“Reg no %d: “, i + 1);

        scanf(“%s“, regs[i]);

    }

    char key[100];

    printf(“Enter registration number to search: “);

    scanf(“%s“, key);

    int low = 0, high = n - 1, found = 0;

    while (low <= high) {

        int mid = (low + high) / 2;

        int cmp = strcmp(key, regs[mid]);

        if (cmp == 0) {

            printf(“Found at index %d (0-based)|n“, mid);

            found = 1;

            break;

        } else if (cmp > 0)

            low = mid + 1;

        else

            high = mid - 1;

    }

    if (!found)

        printf(“Not found|n“);

  return 0;

}
```

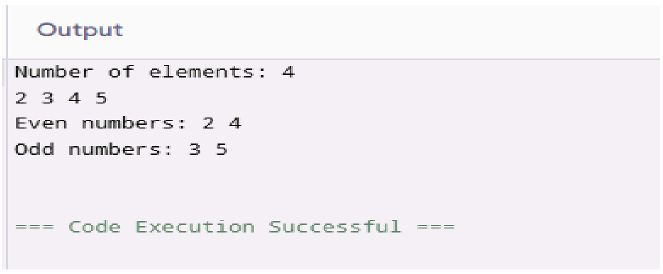


Output

```
Enter the number of registration numbers: 6
Enter 6 registration numbers in ascending order:
101 202 303 404 505 606
Enter the registration number to search: 505
Registration number 505 found at position 5.


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

EXP NO 6:- . Implement a C Program Given array print odd and even values

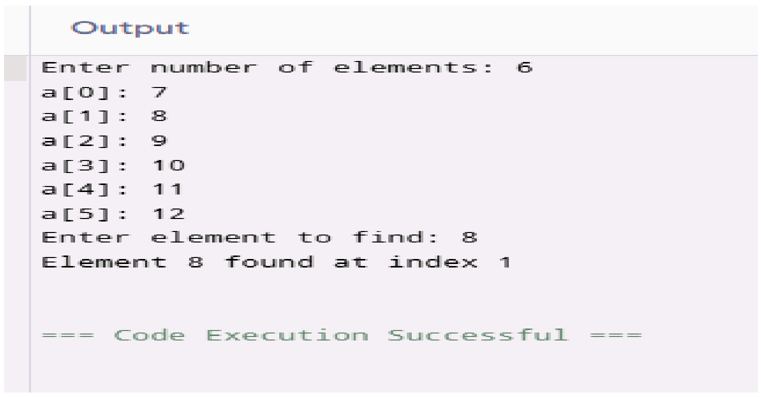
```
#include <stdio.h>

int main() {

    int n;

    printf("Number of elements: ");

    if (scanf("%d", &n) != 1 || n <= 0) return 0;

    int a[n];

    for (int i = 0; i < n; ++i) scanf("%d", &a[i]);

    printf("Even numbers: ");

    for (int i = 0; i < n; ++i) if (a[i] % 2 == 0) printf("%d ", a[i]);

    printf("\nOdd numbers: ");

    for (int i = 0; i < n; ++i) if (a[i] % 2 != 0) printf("%d ", a[i]);

    printf("\n");

    return 0;

}
```


Output

```
Number of elements: 4
2 3 4 5
Even numbers: 2 4
Odd numbers: 3 5


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

EXP NO 5:-Implement a C Program Identify location of element in given array

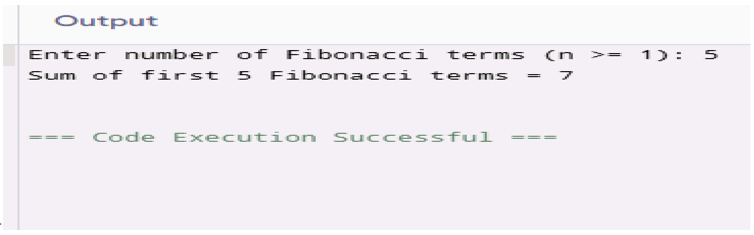
```
#include <stdio.h>



int main() {

    int n;

    printf("Enter number of elements: ");

    if (scanf("%d", &n) != 1 || n <= 0) return 0;

    int a[n];

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

        printf("a[%d]: ", i);
        scanf(“%d“, &a[i]);
    }
    int key;
    printf(“Enter element to find: “);
    scanf(“%d“, &key);
    int found = 0;
    for (int i = 0; i < n; ++i) {
        if (a[i] == key) {
            printf(“Element %d found at index %d\n“, key, i);
            found = 1;
            break; // remove break to print all occurrences
        }
    }
    if (!found) printf(“Element not found\n“);
    return 0;
}
```



```
Output
Enter number of elements: 6
a[0]: 7
a[1]: 8
a[2]: 9
a[3]: 10
a[4]: 11
a[5]: 12
Enter element to find: 8
Element 8 found at index 1


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

EXP NO 7:- Implement a C Program sum of Fibonacci Series

```
#include <stdio.h>
int main() {
    int n;
    printf(“Enter number of Fibonacci terms (n >= 1): “);
    if (scanf(“%d“, &n) != 1 || n < 1) return 0;
    long long a = 0, b = 1;

    long long sum = 0;

    if (n >= 1) sum += a; // include F0 if desired; if you want starting at F1 change accordingly

    if (n >= 2) sum += b;

    for (int i = 3; i <= n; ++i) {

        long long c = a + b;

        sum += c;

        a = b; b = c;

    }

    printf("Sum of first %d Fibonacci terms = %lld\n", n, sum);

    return 0;
```

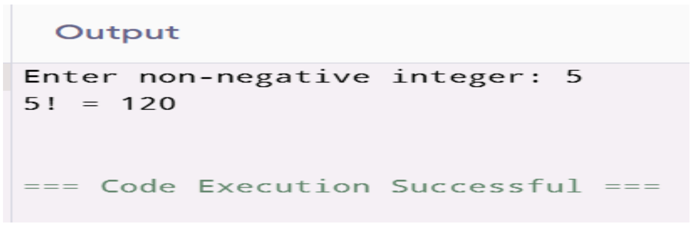


```
Output
Enter number of Fibonacci terms (n >= 1): 5
Sum of first 5 Fibonacci terms = 7

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

```
}
```

EXP NO 8:- Implement a C Program for Finding factorial of a number

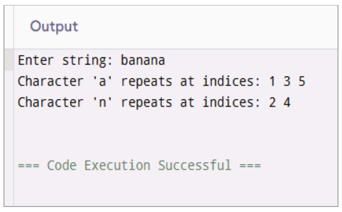
```
#include <stdio.h>

unsigned long long factorial(unsigned int n) {

    unsigned long long res = 1;

    for (unsigned int i = 2; i <= n; ++i) res *= i;

    return res;

}

int main() {

    unsigned int n;

    printf("Enter non-negative integer: ");

    if (scanf("%u", &n) != 1) return 0;

    printf("%u! = %llu\n", n, factorial(n));

    return 0;

}
```



```
Output
Enter non-negative integer: 5
5! = 120


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

EXP NO 9:- Implement a C Program to Print the index of repeated characters given in an array

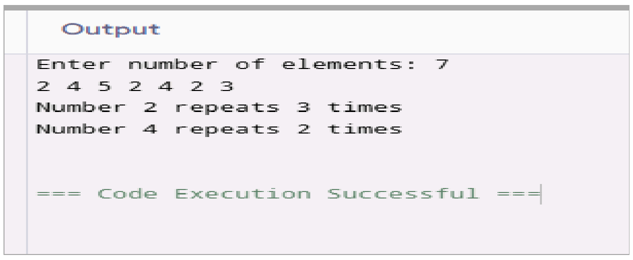
```
#include <stdio.h>

#include <string.h>

int main() {

  char s[1024];

  printf("Enter string: ");

  if (!fgets(s, sizeof(s), stdin)) return 0;

  s[strcspn(s, "\n")] = 0;

  int len = strlen(s);

  int printed[256] = {0};

  for (int ch = 0; ch < 256; ++ch) {

    int count = 0;

    for (int i = 0; i < len; ++i) if ((unsigned char)s[i] == ch) ++count;

    if (count > 1) {

      printf("Character '%c' repeats at indices: ", (char)ch);

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

        if ((unsigned char)s[i] == ch) printf("%d ", i);

      }

      printf("\n");

    }

  }

  return 0;

}
```



```
Output
Enter string: banana
Character 'a' repeats at indices: 1 3 5
Character 'n' repeats at indices: 2 4


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

EXP NO 10:- Implement a C Program to Print the frequently repeated numbers count from an array

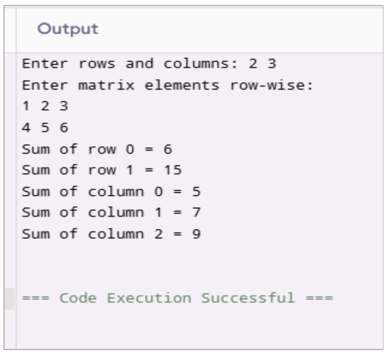
```
#include <stdio.h>

#include <stdlib.h>

int cmp_int(const void *a, const void *b) {

    return (*(int*)a) - (*(int*)b);

}

int main() {

    int n;

    printf("Enter number of elements: ");

    if (scanf("%d", &n) != 1 || n <= 0) return 0;

    int *a = malloc(n * sizeof(int));

    for (int i = 0; i < n; ++i) scanf("%d", &a[i]);

  qsort(a, n, sizeof(int), cmp_int);

    int i = 0;

    int found_any = 0;

    while (i < n) {

        int j = i + 1;

        while (j < n && a[j] == a[i]) ++j;

        int count = j - i;

        if (count > 1) {

            printf("Number %d repeats %d times|n", a[i], count);

            found_any = 1;

        }

        i = j;

    }

    if (!found_any) printf("No repeated numbers|n");

    free(a);

    return 0;

}
```



```
Enter number of elements: 7
2 4 5 2 4 2 3
Number 2 repeats 3 times
Number 4 repeats 2 times


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

EXP NO 11:- . Implement a C Program to perform Sum of row and column in an Array

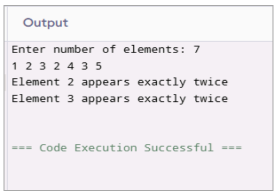
```
#include <stdio.h>

#include <stdlib.h>

int main() {

    int r, c;

    printf("Enter rows and columns: ");

    if (scanf("%d %d", &r, &c) != 2 || r <= 0 || c <= 0) return 0;

    int **mat = malloc(r * sizeof(int*));

    for (int i = 0; i < r; ++i) mat[i] = malloc(c * sizeof(int));

    printf("Enter matrix elements row-wise:\n");

    for (int i = 0; i < r; ++i)

        for (int j = 0; j < c; ++j)

            scanf("%d", &mat[i][j]);

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

        long long sum = 0;

        for (int j = 0; j < c; ++j) sum += mat[i][j];

        printf("Sum of row %d = %lld\n", i, sum);

    }

    for (int j = 0; j < c; ++j) {

        long long sum = 0;

        for (int i = 0; i < r; ++i) sum += mat[i][j];

        printf("Sum of column %d = %lld\n", j, sum);

    }

    for (int i = 0; i < r; ++i) free(mat[i]);

    free(mat);

    return 0;

}
```


Output

```
Enter rows and columns: 2 3
Enter matrix elements row-wise:
1 2 3
4 5 6
Sum of row 0 = 6
Sum of row 1 = 15
Sum of column 0 = 5
Sum of column 1 = 7
Sum of column 2 = 9


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

EXP NO 12:- . Implement a C Program to check for Elements repeated twice – Array

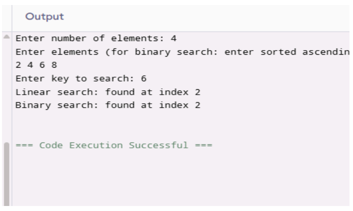
```
#include <stdio.h>

#include <stdlib.h>

int cmp_int(const void *a, const void *b) { return (*(int*)a) - (*(int*)b); }

int main() {

    int n;

    printf("Enter number of elements: ");

    if (scanf("%d", &n) != 1 || n <= 0) return 0;

    int *a = malloc(n * sizeof(int));

    for (int i = 0; i < n; ++i) scanf("%d", &a[i]);

    qsort(a, n, sizeof(int), cmp_int);

  int i = 0;

    int found = 0;

    while (i < n) {

        int j = i + 1;

        while (j < n && a[j] == a[i]) ++j;

        int count = j - i;

        if (count == 2) {

            printf("Element %d appears exactly twice|n", a[i]);

            found = 1;

        }

        i = j;

    }

    if (!found) printf("No element appears exactly twice|n");

    free(a);
    return 0;

}
```



```
Output
Enter number of elements: 7
1 2 3 2 4 3 5
Element 2 appears exactly twice
Element 3 appears exactly twice


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

EXP NO 13:- A Implement a C Program for array search - linear and binary search

```
#include <stdio.h>

int linear_search(int a[], int n, int key) {

    for (int i = 0; i < n; ++i) if (a[i] == key) return i;

    return -1;

}

int binary_search(int a[], int n, int key) {

    int lo = 0, hi = n - 1;

    while (lo <= hi) {

        int mid = lo + (hi - lo) / 2;

        if (a[mid] == key) return mid;

        else if (a[mid] < key) lo = mid + 1;

        else hi = mid - 1;

    }

    return -1;

}

int main() {

    int n;

    printf("Enter number of elements: ");

    if (scanf("%d", &n) != 1 || n <= 0) return 0;

    int a[n];

    printf("Enter elements (for binary search: enter sorted ascending array):\n");

    for (int i = 0; i < n; ++i) scanf("%d", &a[i]);

    int key;

    printf("Enter key to search: ");

    scanf("%d", &key);
    int li = linear_search(a, n, key);

    if (li >= 0) printf("Linear search: found at index %d\n", li);

    else printf("Linear search: not found\n");

    int bi = binary_search(a, n, key);

    if (bi >= 0) printf("Binary search: found at index %d\n", bi);

    else printf("Binary search: not found (ensure array is sorted)\n");

    return 0;

}
```

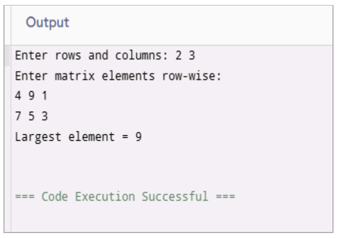


Output

```
Enter number of elements: 4
Enter elements (for binary search: enter sorted ascendin
2 4 6 8
Enter key to search: 6
Linear search: found at index 2
Binary search: found at index 2


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

EXP NO 14:- Implement a C Program for Given 2 D matrix to print largest element

```
#include <stdio.h>

#include <limits.h>

#include <stdlib.h>

int main() {

    int r, c;

    printf("Enter rows and columns: ");

    if (scanf("%d %d", &r, &c) != 2 || r <= 0 || c <= 0) return 0;

    int maxv = INT_MIN;

    printf("Enter matrix elements row-wise:\n");

    for (int i = 0; i < r; ++i)

        for (int j = 0; j < c; ++j) {

            int v;

            scanf("%d", &v);

            if (v > maxv) maxv = v;

        }

    printf("Largest element = %d\n", maxv);

    return 0;

}
```

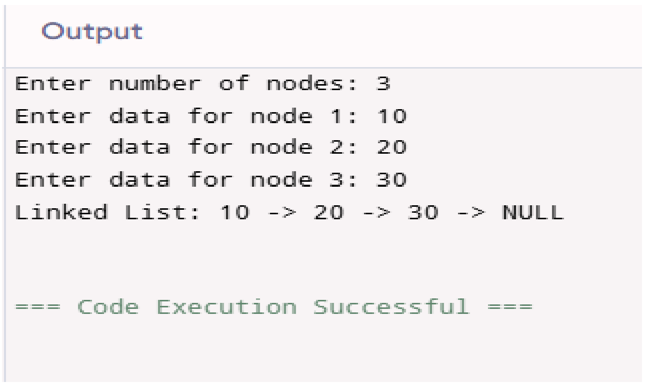


```
Enter rows and columns: 2 3
Enter matrix elements row-wise:
4 9 1
7 5 3
Largest element = 9


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

EXP NO 15: 3D MULTIPLICATION,ADD,SUB,TRANSPOSE

EXP NO 16: C PROGRAM TO IMPLEMENT A SINGLE LINKED LIST

```
#include <stdio.h>

#include <stdlib.h>

struct Node {

    int data;

    struct Node *next;

};

int main() {

    struct Node *head = NULL, *temp, *newnode;

    int n, value;

    printf("Enter number of nodes: ");

    scanf("%d", &n);

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

        newnode = (struct Node *)malloc(sizeof(struct Node));

        printf("Enter data for node %d: ", i + 1);

        scanf("%d", &value);

        newnode->data = value;

        newnode->next = NULL;

        if (head == NULL)

            head = newnode;

        else {

            temp = head;

            while (temp->next)

                temp = temp->next;

            temp->next = newnode;

        }

    }
printf("Linked List: ");
    temp = head;
    while (temp) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
    return 0;
}
```

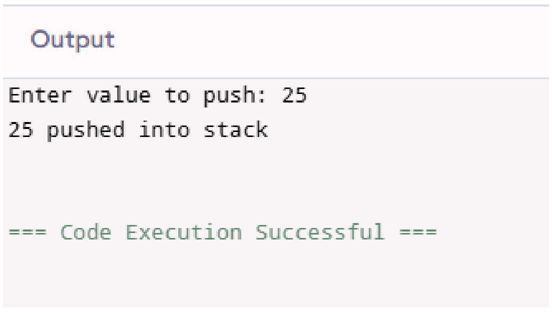


Output

```
Enter number of nodes: 3
Enter data for node 1: 10
Enter data for node 2: 20
Enter data for node 3: 30
Linked List: 10 -> 20 -> 30 -> NULL


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

EXP NO17: WRITE A C PROGRAM TO IMPLEMENT STACK OPERATION IN ARRAY[PUSH]

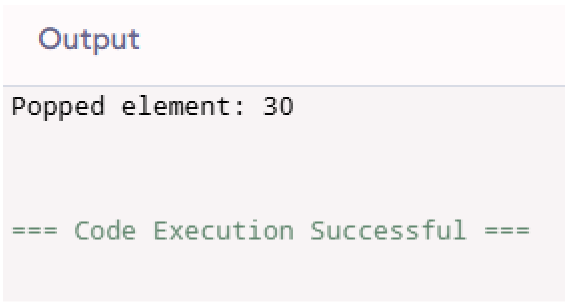
```
#include <stdio.h>
#define MAX 5
int stack[MAX], top = -1;
void push(int val) {
    if (top == MAX - 1)
        printf("Stack Overflow\n");
    else {
        top++;
        stack[top] = val;
        printf("%d pushed into stack\n", val);
    }
}
int main() {
    int val;
    printf("Enter value to push: ");
scanf("%d", &val);
    push(val);
    return 0;
}
```



```
Output
Enter value to push: 25
25 pushed into stack


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

EXP NO18: WRITE A C PROGRAM TO IMPLEMENT STACK OPERATIONIN ARRAY[POP]

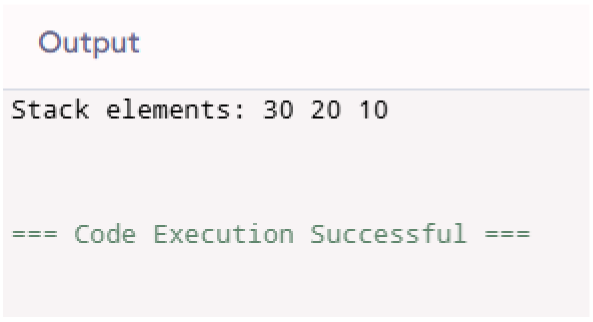
```
#include <stdio.h>
#define MAX 5
int stack[MAX] = {10, 20, 30}, top = 2;

void pop() {
    if (top == -1)
        printf("Stack Underflow|n");
    else
        printf("Popped element: %d|n", stack[top--]);
}

int main() {
    pop();
    return 0;
}
```



```
Output
Popped element: 30


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

EXP NO19: WRITE A C PROGRAM TO IMPLEMENT STACK OPERATION IN ARRAY[DISPLAY]

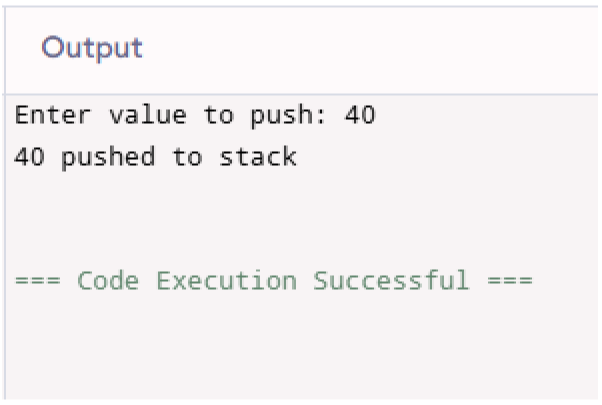
```
#include <stdio.h>

#define MAX 5

int stack[MAX] = {10, 20, 30}, top = 2;


void display() {

    if (top == -1)

        printf("Stack is empty|n");

    else {

        printf("Stack elements: ");

        for (int i = top; i >= 0; i--)

            printf("%d ", stack[i]);

        printf("|n");

    }

}


int main() {

    display();

    return 0;

}
```


Output

```
Stack elements: 30 20 10


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

EXP NO20: WRITE A C PROGRAM TO IMPLEMENT STACK OPERATION IN LINKED LIST[PUSH]

```
#include <stdio.h>

#include <stdlib.h>

struct Node {

    int data;

 struct Node *next;
};

struct Node *top = NULL;

void push(int val) {

    struct Node *newnode = (struct Node *)malloc(sizeof(struct Node));

    newnode->data = val;

    newnode->next = top;

    top = newnode;

    printf("%d pushed to stack\n", val);

}

int main() {

    int val;

    printf("Enter value to push: ");

    scanf("%d", &val);

    push(val);

    return 0;

}
```



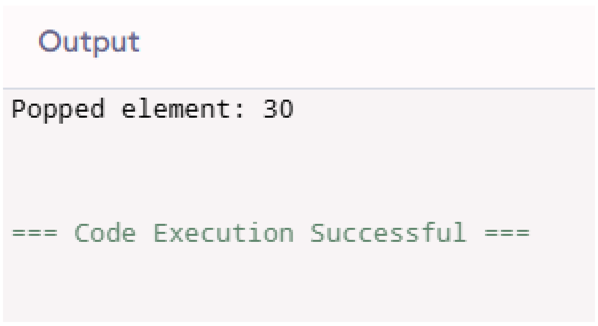
Output

```
Enter value to push: 40
40 pushed to stack


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

EXP NO21: WRITE A C PROGRAM TO IMPLEMENT STACK OPERATION IN LINKED LIST[POP]

```
#include <stdio.h>

#include <stdlib.h>

struct Node {

    int data;

    struct Node *next;

};
```

```
struct Node *top = NULL;
void push(int val) {
    struct Node *newnode = (struct Node *)malloc(sizeof(struct Node));
    newnode->data = val;
    newnode->next = top;
    top = newnode;
}
void pop() {
    if (top == NULL)
        printf("Stack Underflow\n");
    else {
        struct Node *temp = top;
        printf("Popped element: %d\n", temp->data);
        top = top->next;
        free(temp);
    }
}
int main() {
    push(10);
    push(20);
    push(30);
    pop();
    return 0;
}
```



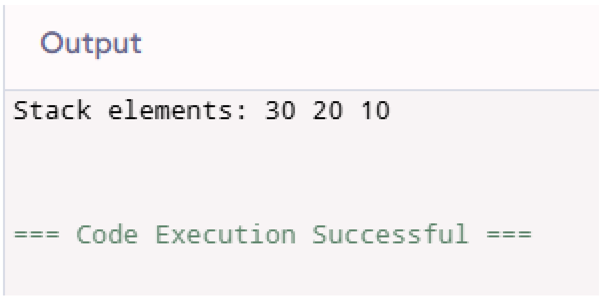
```
Popped element: 30


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

EXP NO22: WRITE A C PROGRAM TO IMPLEMENT STACK OPERATION IN LINKED LIST[DISPLAY]

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

```
#include <stdlib.h>
struct Node {
    int data;
    struct Node *next;
};
struct Node *top = NULL;
void push(int val) {
    struct Node *newnode = (struct Node *)malloc(sizeof(struct Node));
    newnode->data = val;
    newnode->next = top;
    top = newnode;
}
void display() {
    if (top == NULL)
        printf(“Stack is empty\n“);
    else {
        struct Node *temp = top;
        printf(“Stack elements: “);
        while (temp) {
            printf(“%d “, temp->data);
            temp = temp->next;
        }
        printf(“\n“);
    }
}
int main() {
    push(10);
    push(20);
    push(30);
    display();
    return 0;
```

}



```
Stack elements: 30 20 10


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

EXP NO 23: Implement a C Program to Print no of nodes in the given linked list

EXP NO 24: Implement a C Program to perform Palindrome using SLL

EXP NO 25: . Implement a C Program to Intersect SLL

EXP NO 26: Implement a C Program to perform linked list – Insertion

EXP NO 27:- Implement a C Program to Reverse – SLL