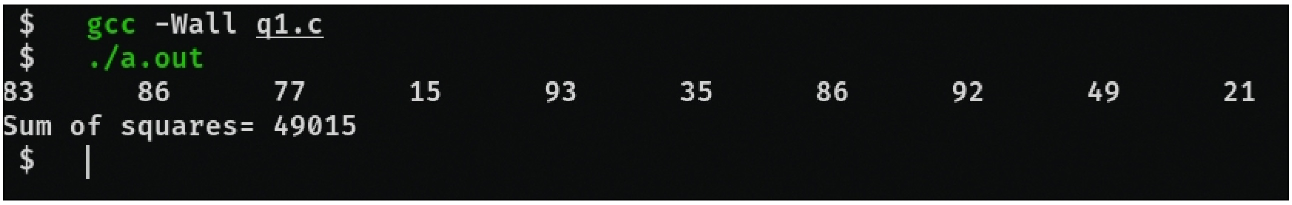
Name: Yashwant Chandrakant Bhosale
MIS: 612303039
Div: SY COMP DIV 1

1. Write a program that calculates the sum of squares of the elements of an integer array of size 10.
code:

```
#include <stdio.h>
#include <stdlib.h>
#define ARRAY_SIZE 10

int sum_squares(int *arr, int len) {
    int sum = 0;
    for(int i = 0; i < len; i++){
        sum += arr[i] * arr[i];
    }
    return sum;
}
void populate(int *arr, int len) {
   for(int i = 0; i < len; i++) {
        arr[i] = rand()%100;
    }
    return;
}
void print(int *arr, int len) {
    for(int i = 0; i < len; i++) {
        printf("%d\t", arr[i]);
    }
    printf("\n");
    return;
}
int main() {
    int arr[ARRAY_SIZE];
    populate(arr, ARRAY_SIZE);
    print(arr, ARRAY_SIZE);
    printf("Sum of squares= %d\n", sum_squares(arr, ARRAY_SIZE));
    return 0;
}
```

Output:

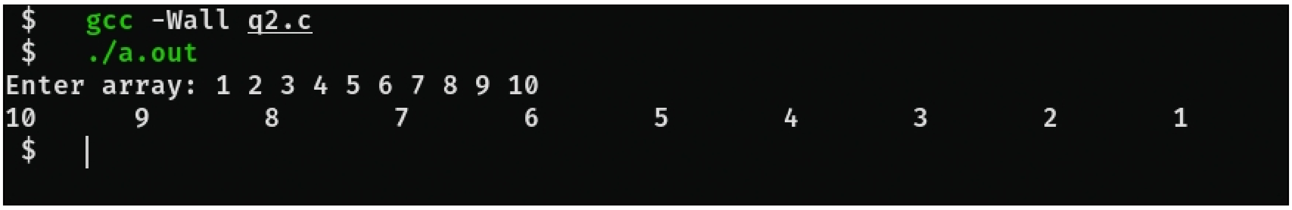


```
 $   gcc -Wall q1.c
 $   ./a.out
83      86      77      15      93      35      86      92      49      21
Sum of squares= 49015
 $   |
```

2. Display array elements in reverse. ie from last to first.
Code:

```
#include <stdio.h>
#include <stdlib.h>
#define ARRAY_SIZE 10
void display_in_reverse(int *arr, int len) {
    for(int j = len-1; j >= 0; j--) {
        printf("%d\t", arr[j]);
    }
    return;
}

void read_array(int *arr, int len) {
    printf("Enter array: ");
    for(int i = 0; i < len; i++) {
        scanf("%d", &arr[i]);
    }
    return;
}

int main() {
    int arr[ARRAY_SIZE];
    read_array(arr, ARRAY_SIZE);
    display_in_reverse(arr, ARRAY_SIZE);
    return 0;
}
```

output:



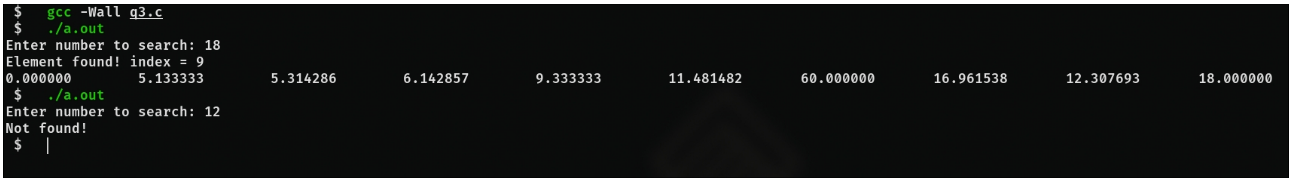
```
 $   gcc -Wall q2.c
 $   ./a.out
Enter array: 1 2 3 4 5 6 7 8 9 10
10      9       8       7       6       5       4       3       2       1
 $   |
```

3. Write a program to search for an element accepted from user in an array of floating- point values of size 50. Display the index if element is found else display message Not Found#include <stdio.h>

```
#include <stdlib.h>
#define ARRAY_SIZE 10
void populate(float *arr, int len) {
    for(int i = 0; i < len; i++) {
        float n = ((float) (rand()%100) * i) / ((float) ((rand() %
50)));
        arr[i] = n;
    }
    return;
}
void print_array(float *arr, int len) {
    for(int i = 0; i < len; i ++) {
        printf("%f\t", arr[i]);
    }
    printf("\n");
    return;
}
```

```
int search(float *arr, int len, float element) {
    for(int i = 0; i < len; i++) {
        if(arr[i] == element)
            return i;
    }
    return -1;
}

int main() {
    float arr[ARRAY_SIZE], n;
    int result;
    populate(arr, ARRAY_SIZE);
    printf("Enter number to search: ");
    scanf("%f", &n);
    result = search(arr, ARRAY_SIZE, n);
    if(result == -1) {
        printf("Not found!\n");
    }
    else{
        printf("Element found! index = %d\n", result);
        print_array(arr, ARRAY_SIZE);
    }
    return 0;
}
```

Output:



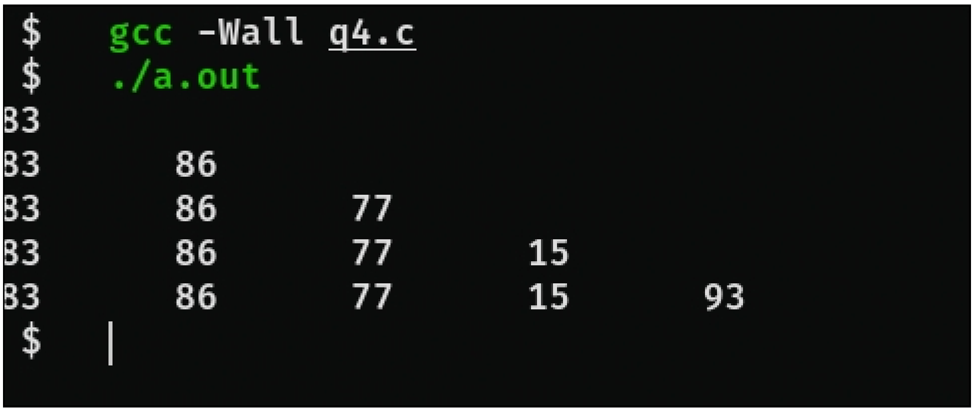
```
 $   gcc -Wall q3.c
 $   ./a.out
Enter number to search: 18
Element found! index = 9
0.000000        5.133333        5.314286        6.142857        9.333333        11.481482       60.000000       16.961538       12.307693       18.000000
 $   ./a.out
Enter number to search: 12
Not found!
 $   |
```

4. Display elements of array in triangle pattern. Use formatting to get a uniform display

```
#include <stdio.h>
#include <stdlib.h>
#define ARRAY_SIZE 5

void populate(int *arr, int len) {
   for(int i = 0; i < len; i++) {
        arr[i] = rand()%100;
    }
    return;
}

void print_triangular(int *arr, int len) {
    printf("%d\n", arr[0]);
    for (int i = 0; i < len-1; i++) {
        for(int j = 0; j <= i+1; j++){
            printf("%d\t", arr[j]);
        }
        printf("\n");
    }
    return;
}
```

```
int main() {
    int arr[ARRAY_SIZE];
    populate(arr, ARRAY_SIZE);
    print_triangular(arr, ARRAY_SIZE);
    return 0;
}
```

Output:

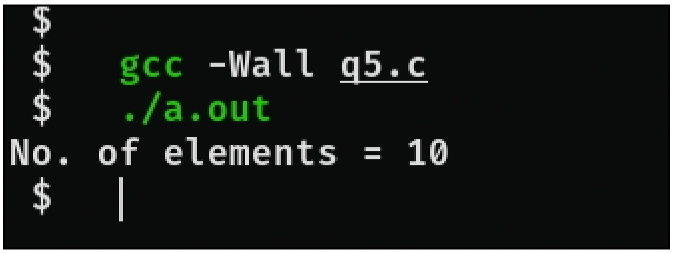


```
 $   gcc -Wall q4.c
 $   ./a.out
83
83      86
83      86      77
83      86      77      15
83      86      77      15      93
 $
```

5. You know size of integer array. Can you find number of elements in it? How?

```
/* If we know size of array in bytes then we may obtain number of
elements by dividing it by sizeof(int) */
#include <stdio.h>
#include <stdlib.h>
#define ARRAY_SIZE 10

int get_size(int arr[], int size) {
    return (size / sizeof(int));
}

int main() {
    int arr[ARRAY_SIZE];
    printf("No. of elements = %d\n", get_size(arr, sizeof(arr)));
    return 0;
}
```

Output:

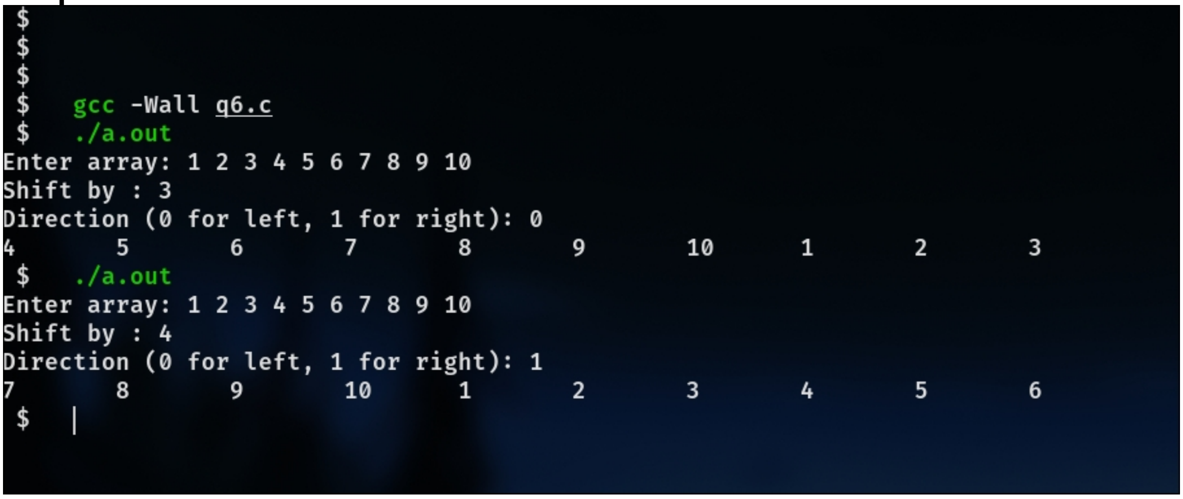


```
 $
 $   gcc -Wall q5.c
 $   ./a.out
No. of elements = 10
 $
```

6. Write C program to shift all elements of an array by n locations to right or left in circular fashion. Take all inputs from user.

```
#include <stdio.h>
#include <stdlib.h>
#define ARRAY_SIZE 10
void shift_by_n(int *arr, int len, int n, int direction) {
    int arr2[len];
    if (direction == 1) {
        // shift to right
        for (int i = 0; i < len; i++) {
            arr2[(i + n) % len] = arr[i];
        }
        for (int i = 0; i < len; i++) {
            arr[i] = arr2[i];
        }
    }else if(direction == 0) {
        // shift to left
        for(int i = 0; i < len; i++){
            arr2[(i - n + len) % len] = arr[i];
        }
        for(int i = 0; i < len; i++){
            arr[i] = arr2[i];
        }
    }
    return;
}

void read_array(int *arr, int len) {
    printf("Enter array: ");
    for (int i = 0; i < len; i++) {
        scanf("%d", &arr[i]);
    }
    return;
}

void print(int *arr, int len) {
    for (int i = 0; i < len; i++) {
        printf("%d\t", arr[i]);
    }
    printf("\n");
    return;
}
int main() {
    int arr[ARRAY_SIZE], n, direction;
    read_array(arr, ARRAY_SIZE);
    printf("Shift by : ");
    scanf("%d", &n);
    printf("Direction (0 for left, 1 for right): ");
    scanf("%d", &direction);
    shift_by_n(arr, ARRAY_SIZE, n, direction);
    print(arr, ARRAY_SIZE);
    return 0;
}
```

Output:

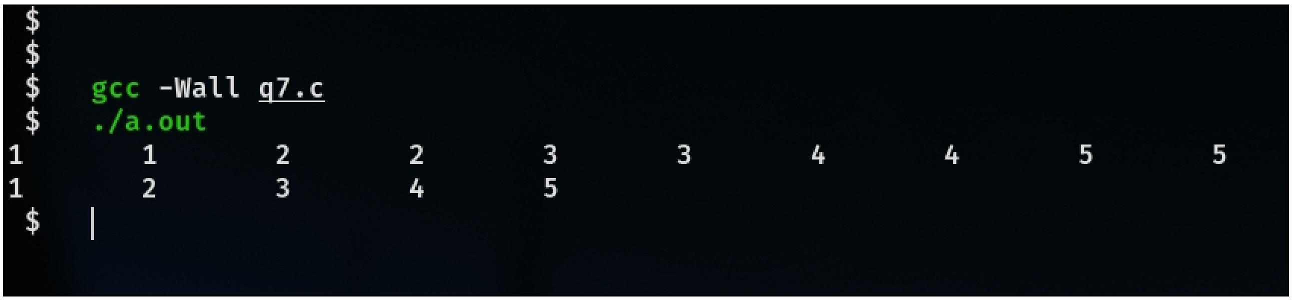


```
 $
 $
 $
 $   gcc -Wall q6.c
 $   ./a.out
Enter array: 1 2 3 4 5 6 7 8 9 10
Shift by : 3
Direction (0 for left, 1 for right): 0
4       5       6       7       8       9       10      1       2       3
 $   ./a.out
Enter array: 1 2 3 4 5 6 7 8 9 10
Shift by : 4
Direction (0 for left, 1 for right): 1
7       8       9       10      1       2       3       4       5       6
 $   |
```

7. Delete all duplicate elements from an array retaining the first occurrence. Note: Array elements cannot be deleted. shift and replace can be done.

```
#include <stdio.h>
void print(int *arr, int len) {
    for(int i = 0; i < len; i++) {
        printf("%d\t", arr[i]);
    }
    printf("\n");
    return;
}
void remove_element(int *arr, int len, int index) {
    for (int i = index; i < len-1; i++) {
        arr[i] = arr[i+1];
    }
    return;
}
int remove_duplicates(int *arr, int len) {
    for(int i = 0; i < len; i++) {
        for(int j = i; j < len; j++) {
            if(arr[i] == arr[j]){
                remove_element(arr, len, j);
                len--;
            }
        }
    }
    return len;
}
int main() {
    int arr[] = {1, 1, 2, 2, 3, 3, 4, 4, 5, 5};
    print(arr, 10);
    int len = remove_duplicates(arr, 10);
    print(arr, len);
    return 0;
}
```

Output:

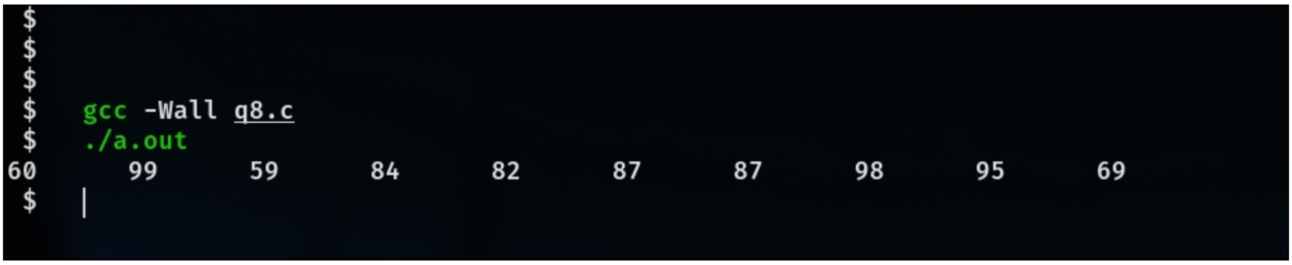


```
 $
 $
 $   gcc -Wall q7.c
 $   ./a.out
1       1       2       2       3       3       4       4       5       5
1       2       3       4       5
 $
```

8. Initialize array of integers with values ranging 50 - 100 both inclusive and display the contents.

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

#define ARRAY_SIZE 10

void populate(int *arr, int len) {
    for (int i = 0; i < len; i++) {
        int n = rand() % 51 + 50;
        arr[i] = n;
    }
    return;
}

void print(int *arr, int len) {
    for(int i = 0; i < len; i++) {
        printf("%d\t", arr[i]);
    }
    printf("\n");
    return;
}


int main() {
    int arr[ARRAY_SIZE];
    populate(arr, ARRAY_SIZE);
    print(arr, ARRAY_SIZE);
    return  0;
}
```

Output:

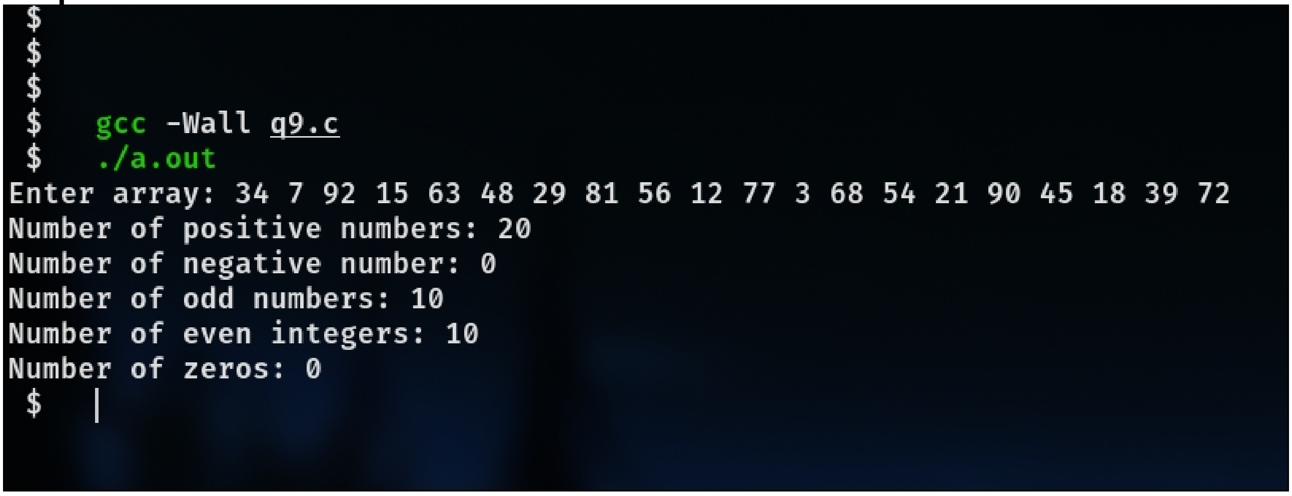


```
 $
 $
 $
 $   gcc -Wall q8.c
 $   ./a.out
60      99      59      84      82      87      87      98      95      69
 $
```

9. Take 20 integer inputs from user and print the following:
number of positive numbers
number of negative numbers
number of odd numbers
number of even numbers
number of 0

```c
#include <stdio.h>
#define ARRAY_SIZE 20

void read_array(int *arr, int len) {
    printf("Enter array: ");
    for(int i = 0; i < len; i++) {
        scanf("%d", &arr[i]);
    }
    return;
}

void print_results(int *arr, int len) {
    int positive=0, negative=0, odd=0, even=0, zeros=0;

    for(int i = 0; i < len; i++) {
        if(arr[i]>0){
            positive++;
        }
        if(arr[i]<0){
            negative++;
        }
        if(arr[i]%2==0){
            even++;
        }
        else{
            odd++;
        }
        if(arr[i] == 0){
            zeros++;
        }
    }
    printf("Number of positive numbers: %d\n", positive);
    printf("Number of negative number: %d\n", negative);
    printf("Number of odd numbers: %d\n", odd);
    printf("Number of even integers: %d\n", even);
    printf("Number of zeros: %d\n", zeros);
    return;
}

int main() {
    int arr[ARRAY_SIZE];
    read_array(arr, ARRAY_SIZE);
    print_results(arr, ARRAY_SIZE);
    return 0;
}
```

Output:

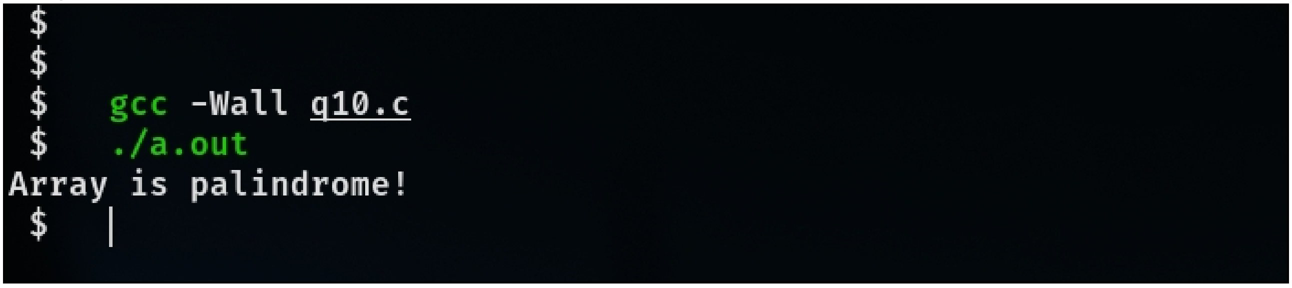


```
 $
 $
 $
 $   gcc -Wall q9.c
 $   ./a.out
Enter array: 34 7 92 15 63 48 29 81 56 12 77 3 68 54 21 90 45 18 39 72
Number of positive numbers: 20
Number of negative number: 0
Number of odd numbers: 10
Number of even integers: 10
Number of zeros: 0
 $   |
```

10. Write a program to check if elements of an array are same or not it read from front or back.

```
#include <stdio.h>

int check_palindrome(int arr[], int len){
    for(int i = 0; i < len / 2 ; i++) {
        if(arr[i] != arr[len - i - 1]) {
            return 0;
        }
    }
    return 1;
}

int main() {
    int arr[] = {2, 3, 15, 15, 3, 2};
    if(check_palindrome(arr, 6)) {
        printf("Array is palindrome!\n");
    }else {
        printf("Array is not Palindrome!");
    }
    return 0;
}
```

output:

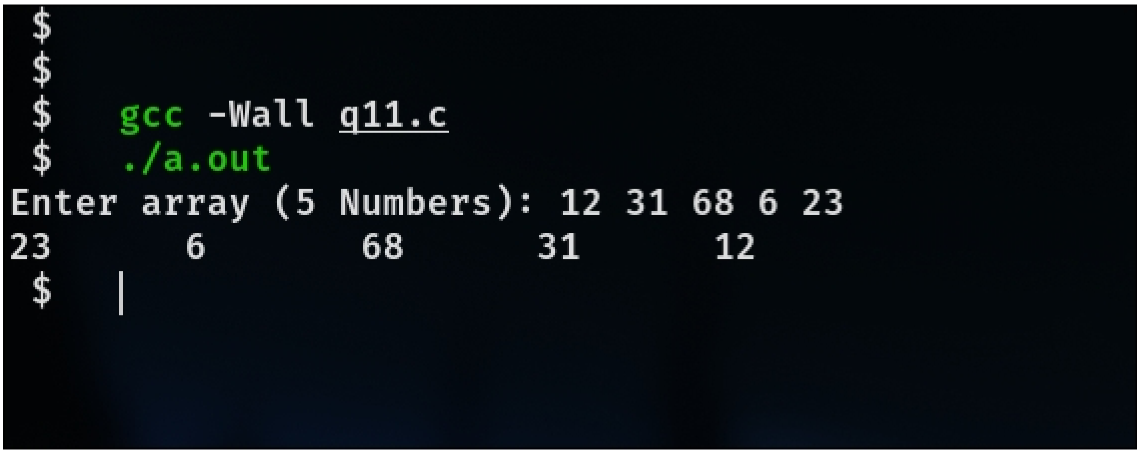


```
 $
 $
 $   gcc -Wall q10.c
 $   ./a.out
Array is palindrome!
 $   |
```

11. Reverse elements of array without using additional array.

```
#include <stdio.h>
#include <stdlib.h>
#define ARRAY_SIZE 5
void print(int *arr, int len) {
    for(int i = 0; i < len; i++) {
        printf("%d\t", arr[i]);
    }
    printf("\n");
    return;
}

void read_array(int *arr, int len) {
    int read_len = 0;
    printf("Enter array (%d Numbers): ", ARRAY_SIZE);
    for(int i = 0; i < len; i++) {
        scanf("%d", &arr[i]);
        read_len++;
    }
    return;
}

void swap(int arr[], int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
    return;
}

void reverse(int *arr, int len) {
    for(int i = 0; i < len/2; i++ ){
        swap(arr, i, len-i-1);
    }
    return;
}


int main() {
    int arr[ARRAY_SIZE];
    read_array(arr, ARRAY_SIZE);
    reverse(arr, ARRAY_SIZE);
    print(arr, ARRAY_SIZE);
    return 0;
}
```

Output:



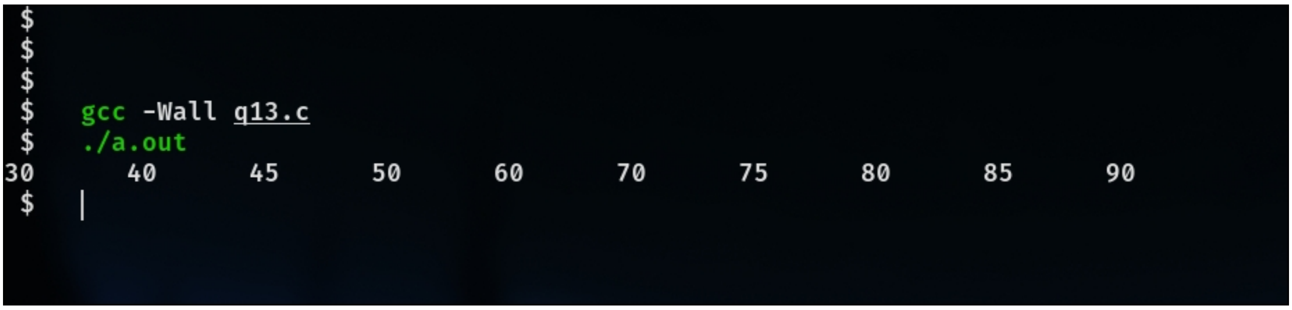
```
 $
 $
 $   gcc -Wall q11.c
 $   ./a.out
Enter array (5 Numbers): 12 31 68 6 23
23      6       68      31      12
 $
```

13. You have 2 arrays of size 5 each having elements in sorted order. Create a new array of 10 having elements of the both the arrays in sorted order

```
#include <stdio.h>
#include <stdlib.h>
void print(int *arr, int len) {
    for(int i = 0; i < len; i++) {
        printf("%d\t", arr[i]);
    }
    printf("\n");
    return;
}
int *merge_and_sort(int arr1[], int len1, int arr2[], int len2) {
    // This function works for two arrays of variable sizes as well
    int len = len1 + len2, i = 0, j = 0, k = 0;
    int *arr3 = (int *) malloc(sizeof(int) * len);
    while (j < len1 && k < len2) {
        // insert the smaller element of the two arrays
        if (arr1[j] <= arr2[k]) {
            arr3[i++] = arr1[j++];
        } else {
            arr3[i++] = arr2[k++];
        }
    }
    // add remaining elements of the whichever array is left
    while (j < len1) {
        arr3[i++] = arr1[j++];
        if(i >= len )
            return arr3;
    }

    while (k < len2) {
        arr3[i++] = arr2[k++];
        if(i >= len )
            return arr3;
    }
    return arr3;
}
```

```
int main() {
    int arr1[] = { 45, 50, 70, 85, 90 };
    int arr2[] = { 30, 40, 60, 75, 80 };
    int *arr3 = merge_and_sort(arr1, 5, arr2, 5);
    print(arr3, 10);
    return 0;
}
```

Output:

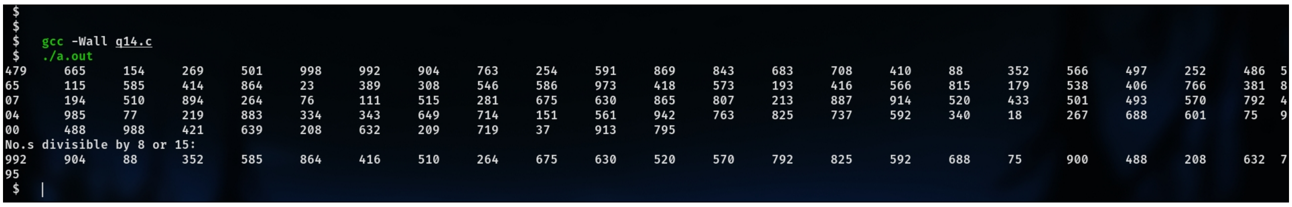


```
$
$
$
$   gcc -Wall q13.c
$   ./a.out
30      40      45      50      60      70      75      80      85      90
$
```

14. Populate an array of size 100 with values generated randomly between 1 to 1000.
Copy all the numbers divisible by 8 or 15 to a new array. Display both arrays.

```
#include <stdio.h>
#include <stdlib.h>
#define ARRAY_SIZE 100

void populate(int *arr, int len) {
    for (int i = 0; i < len; i++) {
        arr[i] = (rand() % 999) + 1;
    }
    return;
}

void print(int *arr, int len) {
    for(int i = 0; i < len; i++) {
        printf("%d\t", arr[i]);
    }
    printf("\n");
    return;
}

int main() {
    int arr[ARRAY_SIZE], arr2[ARRAY_SIZE], len = 0;
    populate(arr, ARRAY_SIZE);

    for(int i = 0; i < ARRAY_SIZE; i++) {
        if(!(arr[i] % 8) || !(arr[i] % 15))
            arr2[len++] = arr[i];
    }
    print(arr, ARRAY_SIZE);
    printf("No.s divisible by 8 or 15: \n");
    print(arr2, len);
    return 0;
}
```

Output:

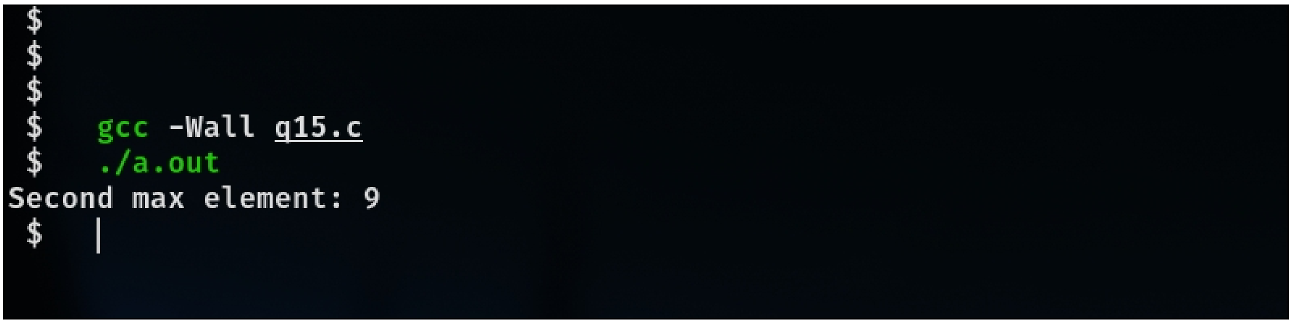


```
 $
 $
 $   gcc -Wall q14.c
 $   ./a.out
479     665     154     269     501     998     992     904     763     254     591     869     843     683     708     410     88      352     566     497     252     486     5
65      115     585     414     864     23      389     308     546     586     973     418     573     193     416     566     815     179     538     406     766     381     8
07      194     510     894     264     76      111     515     281     675     630     865     807     213     887     914     520     433     501     493     570     792     4
04      985     77      219     883     334     343     649     714     151     561     942     763     825     737     592     340     18      267     688     601     75      9
00      488     988     421     639     208     632     209     719     37      913     795
No.s divisible by 8 or 15:
992     904     88      352     585     864     416     510     264     675     630     520     570     792     825     592     688     75      900     488     208     632     7
95
 $   |
```

15. Write code to find second largest element in a 1D Array

```
#include <stdio.h>

int main() {
    int arr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, max = 0, second_max
= 0;
    for(int i = 0; i < 10; i++) {
        if(arr[i] > max) {
            second_max = max;
            max = arr[i];
        } else if(arr[i] > second_max) {
            second_max = arr[i];
        }
    }
    printf("Second max element: %d\n", second_max);
    return 0;
}
```

Output:



```
 $
 $
 $
 $   gcc -Wall q15.c
 $   ./a.out
Second max element: 9
 $   |
```