

ALGO LAB 2

Lab 2 Arrays – 13/08/2020

- 1) Assume A is an array of size n. A group $S = \{ Y_k \in A : 1 \leq k \leq m \}$ is said to be a “group of even numbers” of size m, if all elements in the group are consecutive elements in the array A and each number is divisible by 2. Otherwise, it is a “group of odd numbers” i.e., all elements in the group are consecutive elements in the array A and each number is not divisible by 2.

Ans.

```
#include <stdio.h>

void printIt(int *arr,int n){
    for (int c = 0; c < n ; c++){
        printf("%d ", arr[c]);
        printf("\n");
    }

    int* even(int *arr,int n) {
        printf("Entered Even section \n-----\n");
        int newEvenArray[20] = {};
        int newEvenArrayCOPY[20] = {};
        int newSize = 0;
        int flag = 0;
        for (int c = 0; c < n ; c++) {
            if(arr[c]%2 == 0) {
                if(newSize == 0) {
                    newEvenArray[newSize] = arr[c];
                    newSize++;
                    flag = 1;
                }
                else if(flag == 1){
                    newEvenArray[newSize] = arr[c];
                    newSize++;
                }
            }
            else{
                if(newSize != 0) {
                    printIt(newEvenArray,newSize);
                }
                newSize = 0;
                flag = 0;
            }
        }
        if(newSize != 0) {
            printIt(newEvenArray,newSize);
        }
    }
}
```

```
    }
    newSize = 0;
    flag = 0;
}

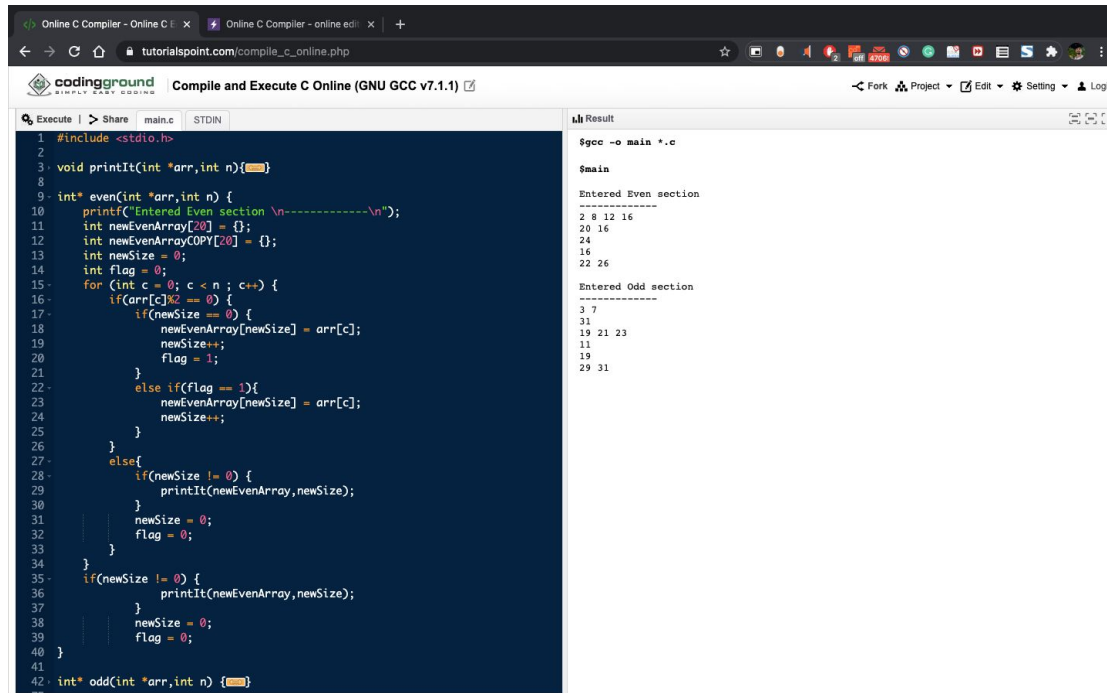
int* odd(int *arr,int n) {
    printf("\nEntered Odd section \n-----\n");
    int newEvenArray[20] = {};
    int newEvenArrayCOPY[20] = {};
    int newSize = 0;
    int flag = 0;
    for (int c = 0; c < n ; c++) {
        if(arr[c]%2 != 0) {
            //printf("%d, ",arr[c]);
            if(newSize == 0 && flag == 0) {
                newEvenArray[newSize] = arr[c];
                newSize++;
                flag = 1;
            }
            else if(flag == 1){
                newEvenArray[newSize] = arr[c];
                newSize++;
            }
        }
        else{
            if(newSize != 0) {
                printf("\n",newEvenArray,newSize);
            }
            newSize = 0;
            flag = 0;
        }
    }
    if(newSize != 0) {
        printf("\n",newEvenArray,newSize);
    }
    newSize = 0;
    flag = 0;
}

int main() {
    int newArr[100] = { 0 };
    int arr1[]={3,7,2,8,12,16,31,20,16,19,21,23,24,11,16,19,22,26,29,31};
    int* arr = arr1;
    int* arr2 = arr1;
    int n = sizeof(arr1)/sizeof(arr1[0]);
    int newPos = 0;
```

```
int dup_n = n;

even(arr,n);
odd(arr2,n);

//printf("%d",n);
return 0;
}
```



The screenshot shows an online C compiler interface. The code editor on the left contains a C program that processes an array of numbers. The program identifies even and odd numbers, groups them, and calculates their LCM. The output window on the right shows the execution results, including the input array, the identified even and odd groups, and the calculated LCM for each group.

```
1 #include <stdio.h>
2
3 void printIt(int *arr,int n){
4
5
6
7
8
9 int* even(int *arr,int n) {
10 printf("Entered Even section \n-----\n");
11 int newEvenArray[20] = {};
12 int newEvenArrayCOPY[20] = {};
13 int newSize = 0;
14 int flag = 0;
15 for (int c = 0; c < n ; c++) {
16 if(arr[c]%2 == 0) {
17 if(newSize == 0) {
18 newEvenArray[newSize] = arr[c];
19 newSize++;
20 flag = 1;
21 }
22 else if(flag == 1){
23 newEvenArray[newSize] = arr[c];
24 newSize++;
25 }
26 }
27 else{
28 if(newSize != 0) {
29 printIt(newEvenArray,newSize);
30 }
31 newSize = 0;
32 flag = 0;
33 }
34 }
35 if(newSize != 0) {
36 printIt(newEvenArray,newSize);
37 }
38 newSize = 0;
39 flag = 0;
40 }
41
42 int* odd(int *arr,int n) {
```

Result

```
$gcc -o main *.c
$main
Entered Even section
-----
2 8 12 16
20 16
24
16
22 26
Entered Odd section
-----
3 7
31
19 21 23
11
19
29 31
```

- 2) Write a program to identify the even groups and odd groups in the given array and calculate the LCM (Least Common Multiple) of each group.

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

```
void printIt(int *arr,int n){
    for (int c = 0; c < n ; c++)
        printf("%d ", arr[c]);
}
```

```
int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}
```

```
int* lcm(int* arr, int n) {
```

```
int ans = arr[0];

for (int i = 1; i < n; i++)
    ans = ((arr[i] * ans) / gcd(arr[i], ans));

printf("\t LCM - %d",ans);
}

int* hcf(int* A, int N){
    int c=gcd(*A,*(A+1));
    int i,g;
    for(i=1;i<N-1;i++)
    {
        g=gcd(c,*(A+1+i));
        c=g;
    }
    printf("\nHCF - %d",c);
}

int* even(int *arr,int n) {
    printf("Entered Even section \n-----\n");
    int newEvenArray[20] = {};
    int newEvenArrayCOPY[20] = {};
    int newSize = 0;
    int flag = 0;
    for (int c = 0; c < n ; c++) {
        if(arr[c]%2 == 0) {
            if(newSize == 0) {
                newEvenArray[newSize] = arr[c];
                newSize++;
                flag = 1;
            }
            else if(flag == 1){
                newEvenArray[newSize] = arr[c];
                newSize++;
            }
        }
        else{
            if(newSize != 0) {
                printf("\n\n");
                hcf(newEvenArray,newSize);
                lcm(newEvenArray,newSize);
                printf("\n\n");
            }
            newSize = 0;
        }
    }
}
```

```
        flag = 0;
    }
}
if(newSize != 0) {
    printIt(newEvenArray,newSize);
    hcf(newEvenArray,newSize);
    lcm(newEvenArray,newSize);
    printf("\n\n");
}
newSize = 0;
flag = 0;
}

int* odd(int *arr,int n) {
    printf("\nEntered Odd section \n-----\n");
    int newEvenArray[20] = {};
    int newEvenArrayCOPY[20] = {};
    int newSize = 0;
    int flag = 0;
    for (int c = 0; c < n ; c++) {
        if(arr[c]%2 != 0) {
            //printf("%d, ",arr[c]);
            if(newSize == 0 && flag == 0) {
                newEvenArray[newSize] = arr[c];
                newSize++;
                flag = 1;
            }
            else if(flag == 1){
                newEvenArray[newSize] = arr[c];
                newSize++;
            }
        }
        else{
            if(newSize != 0) {
                printIt(newEvenArray,newSize);
                hcf(newEvenArray,newSize);
                lcm(newEvenArray,newSize);
                printf("\n\n");
            }
            newSize = 0;
            flag = 0;
        }
    }
    if(newSize != 0) {
        printIt(newEvenArray,newSize);
        hcf(newEvenArray,newSize);
```

```

        lcm(newEvenArray,newSize);
        printf("\n\n");
    }
    newSize = 0;
    flag = 0;
}

int main() {
    int newArr[100] = { 0 };
    int arr1[]={3,7,2,8,12,16,31,20,16,19,21,23,24,11,16,19,22,26,29,31};
    int* arr = arr1;
    int* arr2 = arr1;
    int n = sizeof(arr1)/sizeof(arr1[0]);
    int newPos = 0;
    int dup_n = n;

    even(arr,n);
    odd(arr2,n);

    //printf("%d",n);
    return 0;
}

```

The screenshot shows a web browser window with the URL `tutorialspoint.com/compile_c_online.php`. The page title is "Compile and Execute C Online (GNU GCC v7.1.1)". The code editor contains the following C code:

```

1 #include <stdio.h>
2
3 void printIt(int *arr,int n){
4
5
6
7
8 int gcd(int a, int b) {
9     if (b == 0)
10        return a;
11    return gcd(b, a % b);
12 }
13
14 int* lcm(int* arr, int n) {
15     int ans = arr[0];
16
17     for (int i = 1; i < n; i++)
18         ans = ((arr[i] * ans) / gcd(arr[i], ans));
19
20     printf("\t LCM - %d",ans);
21 }
22
23 int* hcf(int* A, int N){
24     int c=gcd(*A,*A+1);
25     int i,g;
26     for(i=1;i<N-1;i++)
27     {
28         g=gcd(c,*A+1+i);
29         c=g;
30     }
31     printf("\nHCF - %d",c);
32 }
33
34 int* even(int *arr,int n) {}
35
36 int* odd(int *arr,int n) {}
37
38 int main() {}

```

The output window shows the following results:

```

$gcc -o main *.c
$main
Entered Even section
2 8 12 16
HCF - 2 LCM - 48
20 16
HCF - 4 LCM - 80
24
HCF - 8 LCM - 24
16
HCF - 16 LCM - 16
22 26
HCF - 2 LCM - 286
Entered Odd section
3 7
HCF - 1 LCM - 21
31
HCF - 1 LCM - 31
19 21 23
HCF - 1 LCM - 9177
11
HCF - 1 LCM - 11
19
HCF - 1 LCM - 19
29 31
HCF - 1 LCM - 899

```

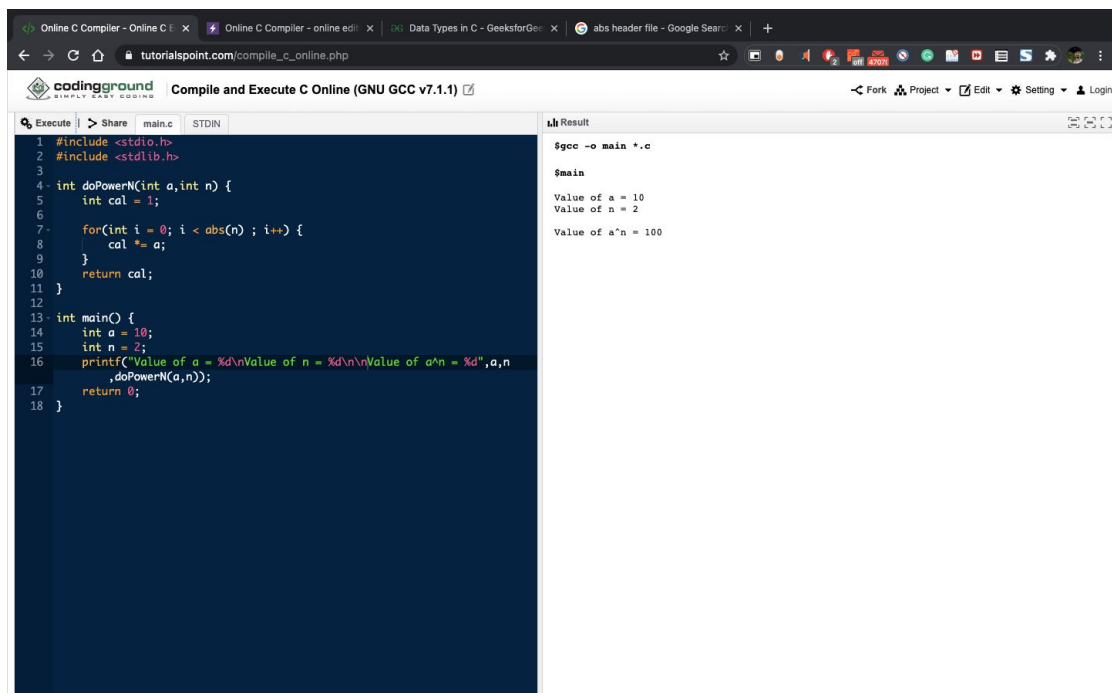
- 3) Let a be an integer and n be a nonnegative integer. Compute a^n . In other words, we ask for a program that does not change the values of a and n but assigns the value of a^n to another variable say cal .

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

int doPowerN(int a,int n) {
    int cal = 1;

    for(int i = 0; i < abs(n) ; i++) {
        cal *= a;
    }
    return cal;
}

int main() {
    int a = 10;
    int n = 2;
    printf("Value of a = %d\nValue of n = %d\n\nValue of a^n = %d",a,n,doPowerN(a,n));
    return 0;
}
```



The screenshot shows a web browser window with the URL `tutorialspoint.com/compile_c_online.php`. The page is titled "Compile and Execute C Online (GNU GCC v7.1.1)". The code editor contains the following C program:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int doPowerN(int a,int n) {
5     int cal = 1;
6
7     for(int i = 0; i < abs(n) ; i++) {
8         cal *= a;
9     }
10    return cal;
11 }
12
13 int main() {
14     int a = 10;
15     int n = 2;
16     printf("Value of a = %d\nValue of n = %d\n\nValue of a^n = %d",a,n
17           ,doPowerN(a,n));
18 }
```

The output window shows the following results:

```
$gcc -o main *.c
$main
Value of a = 10
Value of n = 2
Value of a^n = 100
```

4) Let $A[1 \dots n]$ be an integer array of size n . Assume that the elements of the array are not in sorted order. Our goal is to re-arrange the elements in sorted order (increasing order of values). We will try to achieve that by using the following logic: We define an operation called SWAP(1,j) where the j th element of the array is swapped with the element located in the 1st position of the array A , i.e. $A[1]$. Use this swap operation as many times as you need to swap any element of the array with the element in the 1st position. After a set of swap operations, the elements of the array should be re-arranged in sorted order. Write a program that will achieve this goal.

```
#include <stdio.h>
int* swap(int *arr,int pos)
{
    int head = arr[0];
    arr[0] = arr[pos];
    arr[pos] = head;
    return arr;
}

void printIt(int *arr,int n){
    for (int c = 0; c < n ; c++)
        printf("%d ", arr[c]);
}

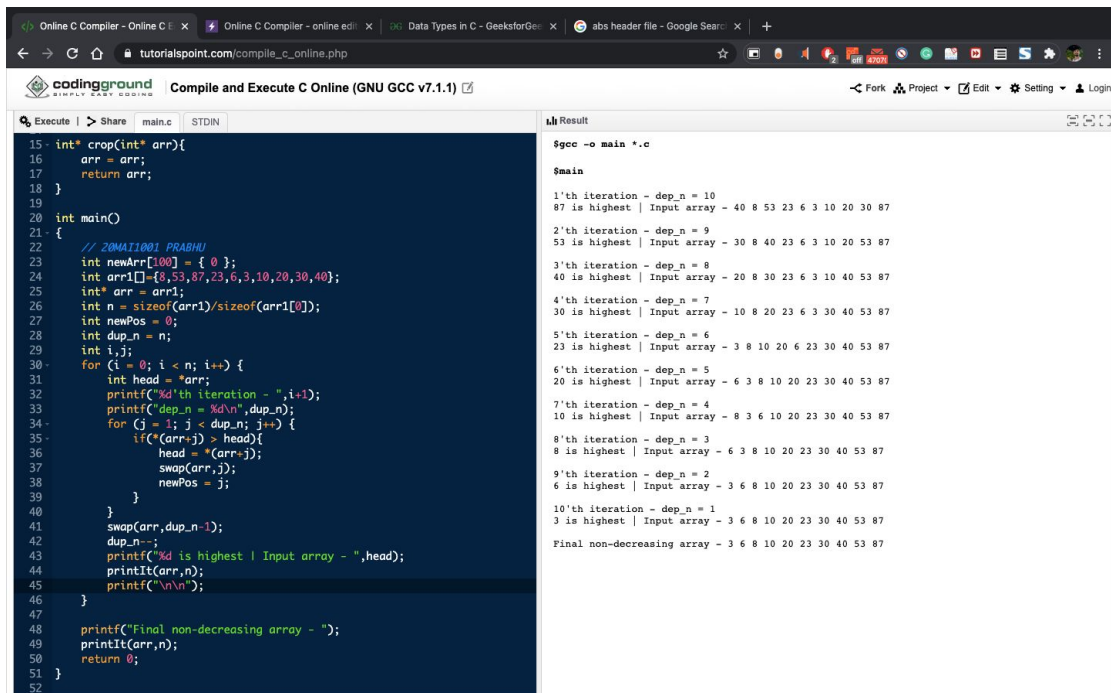
int* crop(int* arr){
    arr = arr;
    return arr;
}

int main()
{
    // 20MAI1001 PRABHU
    int newArr[100] = { 0 };
    int arr1[]={8,53,87,23,6,3,10,20,30,40};
    int* arr = arr1;
    int n = sizeof(arr1)/sizeof(arr1[0]);
    int newPos = 0;
    int dup_n = n;
    int i,j;
    for (i = 0; i < n; i++) {
        int head = *arr;
        printf("%d'th iteration - ",i+1);
        printf("dup_n = %d\n",dup_n);
        for (j = 1; j < dup_n; j++) {
            if(*(arr+j) > head){
```



```
        head = *(arr+j);
        swap(arr,j);
        newPos = j;
    }
}
swap(arr,dup_n-1);
dup_n--;
printf("%d is highest | Input array - ",head);
printf(arr,n);
printf("\n\n");
}

printf("Final non-decreasing array - ");
printf(arr,n);
return 0;
}
```



The screenshot shows a web browser window with the URL `tutorialspoint.com/compile_c_online.php`. The page title is "Compile and Execute C Online (GNU GCC v7.1.1)". The code editor on the left contains the following C code:

```
15 int* crop(int* arr){
16     arr = arr;
17     return arr;
18 }
19
20 int main()
21 {
22     // 20MAI1001 PRABHU
23     int newArr[100] = { 0 };
24     int arr1[]={8,53,87,23,6,3,10,20,30,40};
25     int* arr = arr1;
26     int n = sizeof(arr1)/sizeof(arr1[0]);
27     int newPos = 0;
28     int dup_n = n;
29     int i,j;
30     for (i = 0; i < n; i++) {
31         int head = *arr;
32         printf("%d'th iteration - ",i+1);
33         printf("dep_n = %d\n",dup_n);
34         for (j = 1; j < dup_n; j++) {
35             if(*(arr+j) > head){
36                 head = *(arr+j);
37                 swap(arr,j);
38                 newPos = j;
39             }
40         }
41         swap(arr,dup_n-1);
42         dup_n--;
43         printf("%d is highest | Input array - ",head);
44         printf(arr,n);
45         printf("\n\n");
46     }
47
48     printf("Final non-decreasing array - ");
49     printf(arr,n);
50     return 0;
51 }
52
```

The output window on the right shows the following results:

```
$gcc -o main *.c
$main
1'th iteration - dep_n = 10
87 is highest | Input array - 40 8 53 23 6 3 10 20 30 87
2'th iteration - dep_n = 9
53 is highest | Input array - 30 8 40 23 6 3 10 20 53 87
3'th iteration - dep_n = 8
40 is highest | Input array - 20 8 30 23 6 3 10 40 53 87
4'th iteration - dep_n = 7
30 is highest | Input array - 10 8 20 23 6 3 30 40 53 87
5'th iteration - dep_n = 6
23 is highest | Input array - 3 8 10 20 6 23 30 40 53 87
6'th iteration - dep_n = 5
20 is highest | Input array - 6 3 8 10 20 23 30 40 53 87
7'th iteration - dep_n = 4
10 is highest | Input array - 8 3 6 10 20 23 30 40 53 87
8'th iteration - dep_n = 3
8 is highest | Input array - 6 3 8 10 20 23 30 40 53 87
9'th iteration - dep_n = 2
6 is highest | Input array - 3 6 8 10 20 23 30 40 53 87
10'th iteration - dep_n = 1
3 is highest | Input array - 3 6 8 10 20 23 30 40 53 87
Final non-decreasing array - 3 6 8 10 20 23 30 40 53 87
```