	COLLEGE OF COMPUTING AND INFORMATION SCIENCES		
	Assignment # 01		
Course Title	Operating System	**Total Marks**	10
Date	26/7/2022	**Class ID**	110084
Student Id	**64091**	**Student Name**	**Hassaan Raheem**

Instructions:
- Copied work and late submission will be marked as ZERO.
- Attach your code and screenshot of your output in this file.
- Submit hardcopy of your assignment on given deadline.

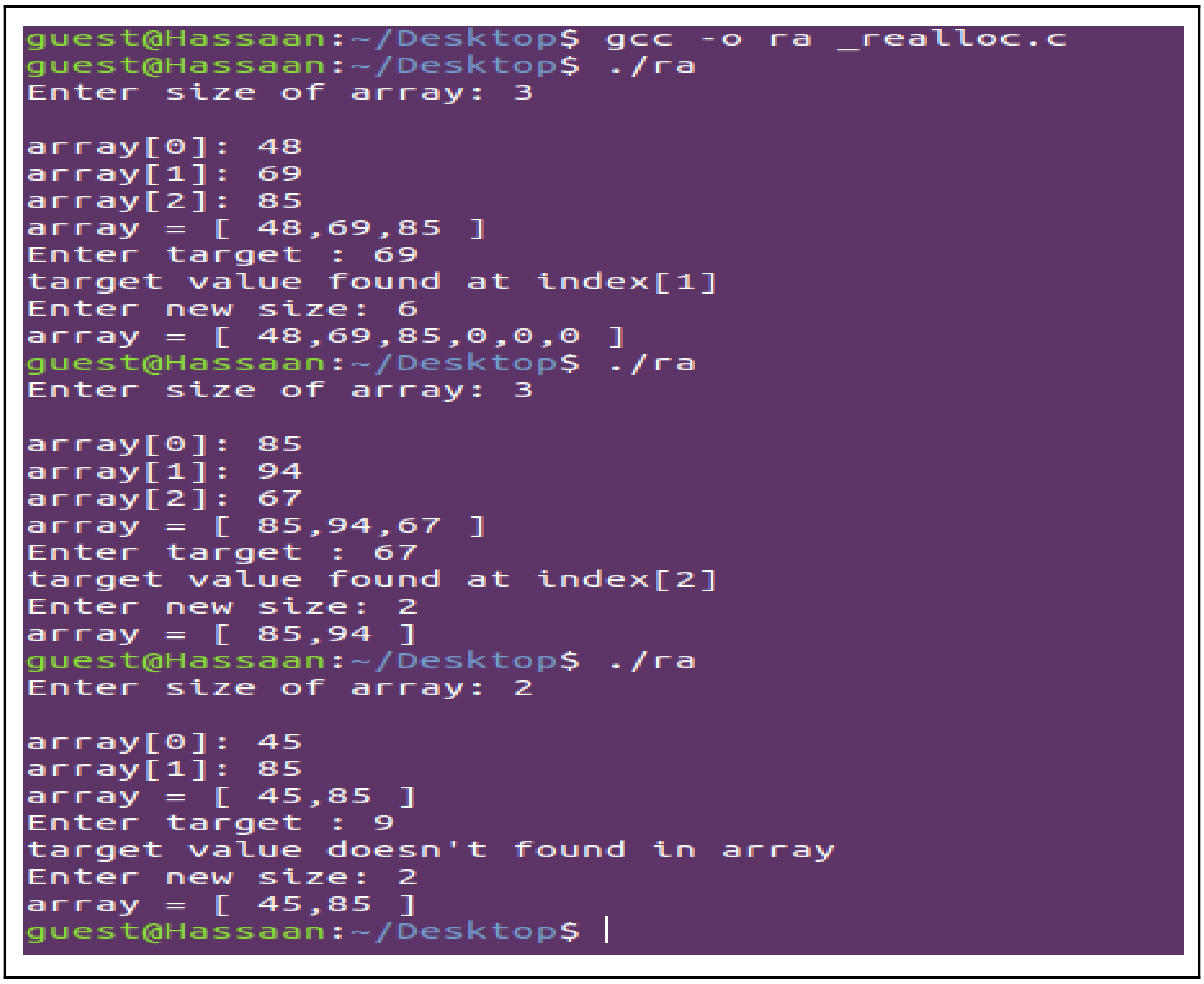
Submission Deadline: 26-07-2022

Question 1:
Write a program that allocate memory for array and print the elements of array (size and elements of array should be user defined), search the number form array using any searching algorithm.
Next, reallocate the size of array and print the array elements. For memory reallocation write down your own **realloc** function using **malloc** and **free** function.
Suppose if your array size is of 5 indexes then the minimum memory required is of 20bytes. After reallocation you can increase or decrease the memory. If the memory size is incremented print 0 at new indexes or if the memory size is decremented, then remove the indexes from array.

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

int gsize; //global variable

//this function will take input and display output at the same
time
int *input_output(int *array , int size){
    printf("\n");
    for(int i=0; i<size; i++){
         printf("array[%d]: ",i);
         scanf("%d",array+i);
    }
    printf("array = [ ");
    for (int j=0; j<size; j++){
         printf("%d,",*(array+j));
    }
    printf("\b ]");
    return array;

}

//this is a linear search function that iterate every element in
the array to check the target value.
void linear_search(int *s , int size){
    int value , i , found = -1 ;
    printf("\nEnter target : ");
    scanf("%d",&value);
    for(i=0 ; i<size ; i++){
          if (value == *(s+i)){
                found = 1;
                break;
          }
    }

    if (found > -1){
          printf("target value found at index[%d]",i);
    }
    else{
          printf("target value doesn't found in array");
    }
    printf("\n");

}

//this is our userdefined _realloc funtion
int *_realloc(int *r ,int size){
    int newsize;
    printf("Enter new size: ");
    scanf("%d",&newsize);
    gsize = newsize;
    // if new size is not same as old size than we need to
rellocate new memory.
    if (newsize != size){
          int *reloc = malloc(newsize*sizeof(int));
          for(int i=0 ; i<newsize ; i++){
                if (i>(size-1)){
                      *(reloc+i) = 0;
                }
                else{
                      *(reloc+i) = *(r+i);
                }
```

```c
        }
        free(r);
        return reloc;
    }
    // if new size is same as old size than we don't need to
reallocate new memory.
    else{
        return r;
    }


}

void display(int *dp , int size){
    printf("\narray = [ ");
    for(int i=0;i<size;i++){
        printf("%d,",*(dp+i));
    }
    printf("\b ]\n");
}

int main()
{
    int size ;
    printf("Enter size of array: ");
    scanf("%d",&size);

    int *array = malloc(size*sizeof(int));

    int *p = input_output(array,size);
    linear_search(p,size);
    int *d =  _realloc(p,size);
    display(d,gsize);

    return 0;
}
```

OUTPUT :



```
guest@Hassaan:~/Desktop$ gcc -o ra _realloc.c
guest@Hassaan:~/Desktop$ ./ra
Enter size of array: 3

array[0]: 48
array[1]: 69
array[2]: 85
array = [ 48,69,85 ]
Enter target : 69
target value found at index[1]
Enter new size: 6
array = [ 48,69,85,0,0,0 ]
guest@Hassaan:~/Desktop$ ./ra
Enter size of array: 3

array[0]: 85
array[1]: 94
array[2]: 67
array = [ 85,94,67 ]
Enter target : 67
target value found at index[2]
Enter new size: 2
array = [ 85,94 ]
guest@Hassaan:~/Desktop$ ./ra
Enter size of array: 2

array[0]: 45
array[1]: 85
array = [ 45,85 ]
Enter target : 9
target value doesn't found in array
Enter new size: 2
array = [ 45,85 ]
guest@Hassaan:~/Desktop$ |
```

Question 2:

Write a program that takes an integer array from user as an input and divide that array into 3 parts. Each part will be given to the thread and finds the smallest value from that. In the end your program should return 3 smallest numbers from array.

Sample Input:

Array: { 5,2,1,4,7,89,6,3,2,5,11,0,5,9,7,6,8,10 }

Sample Output:

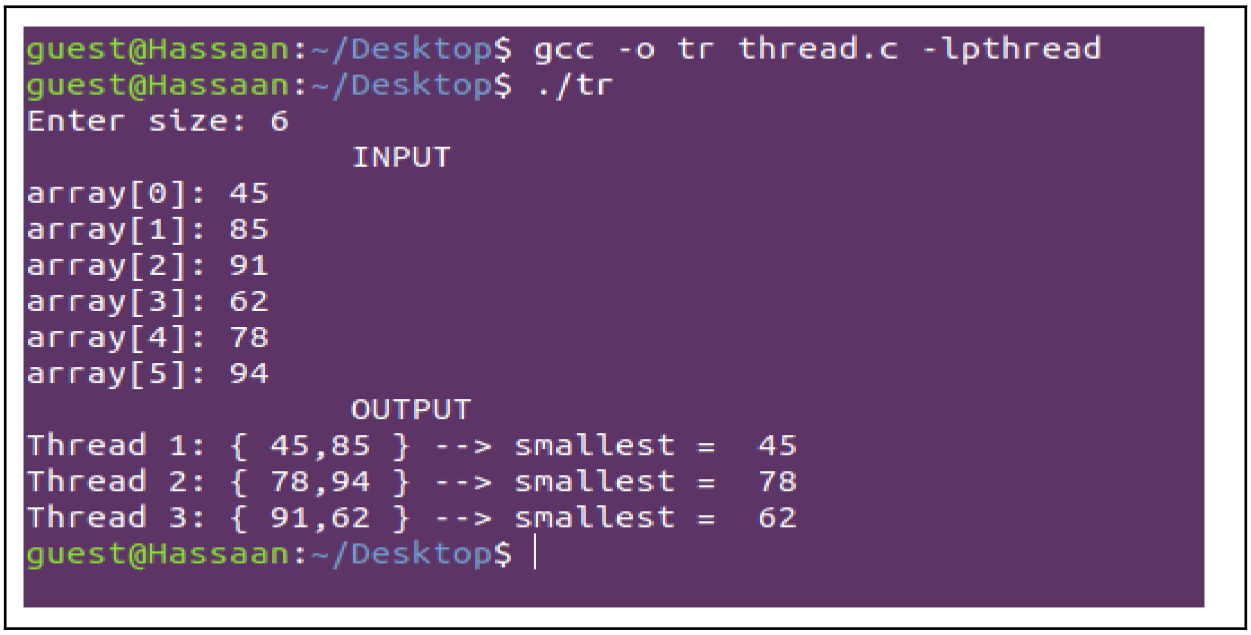
```
Thread 1: {5, 2, 1, 4, 7, 89} -> smallest = 1
Thread 2: {6, 3, 2, 5, 11, 0} -> smallest = 0
Thread 3: {5, 9, 7, 6, 8, 10} -> smallest = 5
```

CODE :

```c
#include <stdio.h> //used for input-output
#include <pthread.h> //used for working on threads
#include <stdlib.h> // used for dynamic memory allocation


//public variables
int sz , x1 , x2 , x3 ,*array, check =1 , tn =1;



//display the elements of each thread.
void display(int dp[] ,int start ,int end ,int tn){
   printf("Thread %d: { ",tn);
        for(int i=start; i<end; i++){
             printf("%d,",array[i]);

        }
        printf("\b }");
}

//find the minimum value from each thread.
void* mini(void* arg){


    int *p = (int *)arg;
    int start = *p;
    int end = start + (sz/3);


    //invoking display function
    display(array,start,end,tn);


    for(int i=start; i<end;i++){
    if (array[start] > array[i]){
        int temp = array[start];
        array[start] = array[i];
        array[i] = temp;
    }
    }
    printf(" --> smallest =  %d\n",array[start]);
    tn ++;

}



void user_input(){
   printf("Enter size: ");
   scanf("%d",&sz);
   printf("\t\tINPUT\n");
   array = malloc(sz*sizeof(int));
   if (sz%3==0 && sz>3){
             for(int i=0;i<sz;i++){
                  printf("array[%d]: ",i);
                  scanf("%d",&array[i]);
             }

        x1 = ((sz/3)*0); // [x1,x1+partition_size] is an intial &
final index of 1st partition.
        x2 = ((sz/3)*1); // [x2,x2+partition_size]] is an intial
& final index of 2nd partition.
        x3 = ((sz/3)*2); // [x3,x3+partition_size] is an initial
& final index of 3rd partition.

   }
   else{
        check = 0;
   }
   printf("\t\tOUTPUT\n");


}

int main()
{

   user_input();

        if (check){
        pthread_t x,y,z;
        pthread_create(&x,NULL,&mini,(void *)&x1); //passing
intial index of 1st partition as func argument.
```

```
        pthread_create(&y,NULL,&mini,(void *)&x2); //passing
intial index of 2nd partition as func argument.
        pthread_create(&z,NULL,&mini,(void *)&x3);
//passinginitial index of 3rd partition as func argument.

        pthread_join(x,NULL);
        pthread_join(y,NULL);
        pthread_join(z,NULL);

   }
   else{
        printf("array size is not multiple of 3\n");
   }



    return 0;
}
```

OUTPUT :



```
guest@Hassaan:~/Desktop$ gcc -o tr thread.c -lpthread
guest@Hassaan:~/Desktop$ ./tr
Enter size: 6
                INPUT
array[0]: 45
array[1]: 85
array[2]: 91
array[3]: 62
array[4]: 78
array[5]: 94
                OUTPUT
Thread 1: { 45,85 } --> smallest =  45
Thread 2: { 78,94 } --> smallest =  78
Thread 3: { 91,62 } --> smallest =  62
guest@Hassaan:~/Desktop$ 
```