ASSIGNMENT-3

```c
/* 2. Design a function with prototype; void sumarr(int a[], int b[], int r[], int
size);that takes 4 parameters,two int arrays as input arguments, 1 array as output
arguments and their effective size respectively to produce a resultant array containing
the sums of corresponding array elements a and b. For example, for the three-element input
arrays 5-1 7 and 2 4-2 , the result would be an array containing 7 3 5 .*/
 #include<stdio.h>

 void sumArr(int a[],int b[],int r[],int size){
    for(int i=0;i<size;i++){
            r[i]=a[i]+b[i];
    }
    printf("Sum Array is: ");
    for(int i=0;i<size;i++){
        printf("%d ",r[i]);
    }
    printf("\n");
 }

 int main(){
    int a[3]={5,-1,7};
    int b[3]={2,4,-2};
    int r[3];
    sumArr(a,b,r,3);
    return 0;
 }
```

```c
/*3. The bubble sort */
 #include<stdio.h>
 void bubbleSort(int arr[],int size){
    for(int i=0;i<size-1;i++){
        for(int j=0;j<size-i-1;j++){
            if(arr[j]>arr[j+1]){
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    printf("After Bubble Sort: ");
    for(int i=0;i<size;i++){
        printf("%d ",arr[i]);
    }
    printf("\n");
 }

 int main(){
    int arr[5]={3,1,4,5,2};
    bubbleSort(arr,5);
    return 0;
 }
```

```c
/*4. Merge Sort without Recursion*/

 #include <stdio.h>
void mergeSort(int arr1[], int m, int arr2[], int n, int merged[]) {
    int i = 0, j = 0, k = 0;
    while (i < m && j < n) {
        if (arr1[i] <= arr2[j]) {
            merged[k++] = arr1[i++]; //take element from arr1
        } else {
            merged[k++] = arr2[j++]; //take element from arr2
        }
    }
    // Add remaining elements from arr1, if any
    while (i < m) {
        merged[k++] = arr1[i++];
    }
    // Add remaining elements from arr2, if any
    while (j < n) {
        merged[k++] = arr2[j++];
    }
    printf("Merged Array: ");
    for (int i = 0; i < m + n; i++) {
        printf("%d ", merged[i]);
    }
    printf("\n");
}
int main() {
    int arr1[] = {1, 3, 5, 7,9};
    int arr2[] = {2, 4, 6, 8, 10};

    int m = sizeof(arr1) / sizeof(arr1[0]);
    int n = sizeof(arr2) / sizeof(arr2[0]);

    int merged[m + n];
    mergeSort(arr1, m, arr2, n, merged);
    return 0;
}
```

```c
/*5.Binary Search*/

#include<stdio.h>
int binarySearch(int arr[],int search,int size){
    int lb=0,ub=size-1,mid;
    while(lb<=ub){
    mid=(lb+ub)/2;

        if(arr[mid]==search){
            return mid;
        }else if(search>mid){
            lb=mid+1;
        }else{
            ub=mid-1;  }   }  return -1; }
```

```c
int main(){
    int arr[5]={1,2,3,4,5};
    int res=binarySearch(arr,5,5);

    if (res!= -1) {
        printf("Element found at index: %d\n", res);
    } else {
        printf("Element not found.\n");
    }
}
```

```c
/* 6. Design a program to find the difference between two sets or arrays. The difference between two sets or arrays: All the elements of the first array that don’t appear in the second array. If array p has the elements 1, 2, 3, 4 and array q has the elements 2, 4, 5, 6 , then the difference between the two arrays, p-q will be 1, 3 .*/
#include <stdio.h>
void main()
{
    int p[] = {1, 2, 3, 4};
    int q[] = {2, 4, 5, 6};
    int size_p = sizeof(p) / sizeof(p[0]);
    int size_q = sizeof(q) / sizeof(q[0]);
    printf("Difference (p - q): ");
    for (int i = 0; i < size_p; i++) {
        int found = 0;
        for (int j = 0; j < size_q; j++) {
            if (p[i] == q[j]){
                found = 1;
                break;
            }
        }
        if (!found)
            printf("%d ", p[i]);
    }
}
```

```c
/* 7.WAP to copy the distinct elements of an int type array to another array.*/
 #include<stdio.h>
 int main(){
    int arr[]={4,7,7,3,2,5,5};
    int size=sizeof(arr)/sizeof(arr[0]);
    for(int i=0;i<size;i++){
        int found=0;
        for(int j=0;j<i;j++){
            if(arr[i]==arr[j]){
                found=1;          }       }
        if(!found){
            printf("%d ",arr[i]);        }          }
  }
```

```c
/* 8. Construct a program to find the occurrence of the first repetitive character in a string. For example, let the string racecar, the program should give the output as The first repititive character in the string racecar is c.*/

#include<stdio.h>
int main(){
    char ch[20]="racecar";
    for(int i=0;i<strlen(ch);i++){
        for(int j=i+1;j<strlen(ch);j++){
            if(ch[i]==ch[j]){
                printf("First repeating element: %c",ch[i]);
                return;
            }        }    }   }
```

```c
/* 9. Design a program to display the count of each character in a string. For example: input string: bintu, output: The count of each character in the string bintu is b-1, i-1, n-1, t-i, u-1.*/
 #include<stdio.h>
 #include<string.h>

 int main(){
    char ch[20]="Shubhangini";
    char occu[256]={0};//char ASCII Range= 0-256
    for(int i=0;i<strlen(ch);i++){
        occu[ch[i]]++;
    }
    printf("Occurence: \n");
    for(int i=0;i<256;i++){
        if(occu[i]>0){
            printf("%c: %d \n",i,occu[i]);
        }
     }
 }
```

```c
/* 10. Write a program to convert a binary number into a hexadecimal number.*/

#include<stdio.h>

int main(){
    int bin=1011;
    int hex=0,i=1;
    while (bin!=0) {
        hex+=(bin%10)*i;
        bin/=10;
        i*=2;
    }
    printf("Hex: %X",hex);
}
```