Array Menu using C

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

struct Array
{
    int *A;
    int size;
    int length;
};

void Display(struct Array arr)
{
    int i;
    printf("\nElements are\n");
    for(i=0;i<arr.length;i++)
        printf("%d ",arr.A[i]);
}

void Append(struct Array *arr,int x)
{
    if(arr->length<arr->size)
        arr->A[arr->length++]=x;

}

void Insert(struct Array *arr,int index,int x)
{
    int i;
    if(index>=0 && index <=arr->length)
    {
        for(i=arr->length;i>index;i--)
            arr->A[i]=arr->A[i-1];
        arr->A[index]=x;
        arr->length++;

    }
}
```

```
int Delete(struct Array *arr,int index)
{
    int x=0;
    int i;

    if(index>=0 && index<arr->length)
    {
        x=arr->A[index];
        for(i=index;i<arr->length-1;i++)
            arr->A[i]=arr->A[i+1];
        arr->length--;
        return x;
    }

    return 0;
}

void swap(int *x,int *y)
{
    int temp;
    temp=*x;
    *x=*y;
    *y=temp;
}

int LinearSearch(struct Array *arr,int key)
{
    int i;
    for(i=0;i<arr->length;i++)
    {
        if(key==arr->A[i])
        {
            swap(&arr->A[i],&arr->A[0]);
            return i;
        }
    }
    return -1;
}
```

```
int BinarySearch(struct Array arr,int key)
{
    int l,mid,h;
    l=0;
    h=arr.length-1;

    while(l<=h)
    {
        mid=(l+h)/2;
        if(key==arr.A[mid])
            return mid;
        else if(key<arr.A[mid])
            h=mid-1;
        else
            l=mid+1;
    }
    return -1;
}

int RBinSearch(int a[],int l,int h,int key)
{
    int mid;

    if(l<=h)
    {
        mid=(l+h)/2;
        if(key==a[mid])
            return mid;
        else if(key<a[mid])
            return RBinSearch(a,l,mid-1,key);
        else
            return RBinSearch(a,mid+1,h,key);
    }
    return -1;
}

int Get(struct Array arr,int index)
{
        if(index>=0 && index<arr.length)
            return arr.A[index];
```

```
    return -1;
}

void Set(struct Array *arr,int index,int x)
{
    if(index>=0 && index<arr->length)
        arr->A[index]=x;
}

int Max(struct Array arr)
{
    int max=arr.A[0];
    int i;
    for(i=1;i<arr.length;i++)
    {
        if(arr.A[i]>max)
            max=arr.A[i];
    }
    return max;
}
int Min(struct Array arr)
{
    int min=arr.A[0];
    int i;
    for(i=1;i<arr.length;i++)
    {
        if(arr.A[i]<min)
            min=arr.A[i];
    }
    return min;
}

int Sum(struct Array arr)
{
    int s=0;
    int i;
    for(i=0;i<arr.length;i++)
        s+=arr.A[i];

    return s;
```

```
}

float Avg(struct Array arr)
{
    return (float)Sum(arr)/arr.length;
}

void Reverse(struct Array *arr)
{
    int *B;
    int i,j;

    B=(int *)malloc(arr->length*sizeof(int));
    for(i=arr->length-1,j=0;i>=0;i--,j++)
        B[j]=arr->A[i];
    for(i=0;i<arr->length;i++)
        arr->A[i]=B[i];

}

void Reverse2(struct Array *arr)
{
    int i,j;
    for(i=0,j=arr->length-1;i<j;i++,j--)
    {
        swap(&arr->A[i],&arr->A[j]);
    }
}

void InsertSort(struct Array *arr,int x)
{
    int i=arr->length-1;
    if(arr->length==arr->size)
        return;
    while(i>=0 && arr->A[i]>x)
    {
        arr->A[i+1]=arr->A[i];
        i--;
    }
    arr->A[i+1]=x;
```

```
    arr->length++;

}

int isSorted(struct Array arr)
{
    int i;
    for(i=0;i<arr.length-1;i++)
    {
        if(arr.A[i]>arr.A[i+1])
            return 0;
    }
    return 1;
}

void Rearrange(struct Array *arr)
{
    int i,j;
    i=0;
    j=arr->length-1;

    while(i<j)
    {
        while(arr->A[i]<0)i++;
        while(arr->A[j]>=0)j--;
        if(i<j)swap(&arr->A[i],&arr->A[j]);
    }

}

struct Array* Merge(struct Array *arr1,struct Array
*arr2)
{
    int i,j,k;
    i=j=k=0;

    struct Array *arr3=(struct Array
*)malloc(sizeof(struct Array));

    while(i<arr1->length && j<arr2->length)
```

```
    {
            if(arr1->A[i]<arr2->A[j])
                arr3->A[k++]=arr1->A[i++];
            else
                arr3->A[k++]=arr2->A[j++];
    }
    for(;i<arr1->length;i++)
        arr3->A[k++]=arr1->A[i];
    for(;j<arr2->length;j++)
        arr3->A[k++]=arr2->A[j];
    arr3->length=arr1->length+arr2->length;
    arr3->size=10;

    return arr3;
}


struct Array* Union(struct Array *arr1,struct Array
*arr2)
{
    int i,j,k;
    i=j=k=0;

    struct Array *arr3=(struct Array
*)malloc(sizeof(struct Array));

    while(i<arr1->length && j<arr2->length)
    {
        if(arr1->A[i]<arr2->A[j])
            arr3->A[k++]=arr1->A[i++];
        else if(arr2->A[j]<arr1->A[i])
            arr3->A[k++]=arr2->A[j++];
        else
        {
            arr3->A[k++]=arr1->A[i++];
            j++;
        }
    }
    for(;i<arr1->length;i++)
        arr3->A[k++]=arr1->A[i];
```

```
    for(;j<arr2->length;j++)
        arr3->A[k++]=arr2->A[j];

    arr3->length=k;
    arr3->size=10;

    return arr3;
}

struct Array* Intersection(struct Array *arr1,struct
Array *arr2)
{
    int i,j,k;
    i=j=k=0;

    struct Array *arr3=(struct Array
*)malloc(sizeof(struct Array));

    while(i<arr1->length && j<arr2->length)
    {
        if(arr1->A[i]<arr2->A[j])
            i++;
        else if(arr2->A[j]<arr1->A[i])
            j++;
        else if(arr1->A[i]==arr2->A[j])
        {
            arr3->A[k++]=arr1->A[i++];
            j++;
        }
    }

    arr3->length=k;
    arr3->size=10;

    return arr3;
}

struct Array* Difference(struct Array *arr1,struct
Array *arr2)
{
```

```
    int i,j,k;
    i=j=k=0;

    struct Array *arr3=(struct Array
*)malloc(sizeof(struct Array));

    while(i<arr1->length && j<arr2->length)
    {
        if(arr1->A[i]<arr2->A[j])
            arr3->A[k++]=arr1->A[i++];
        else if(arr2->A[j]<arr1->A[i])
            j++;
        else
        {
            i++;
            j++;
        }
    }
    for(;i<arr1->length;i++)
        arr3->A[k++]=arr1->A[i];


    arr3->length=k;
    arr3->size=10;

    return arr3;
}



int main()
{
    struct Array arr1;
    int ch;
    int x,index;

    printf("Enter Size of Array");
    scanf("%d",&arr1.size);
    arr1.A=(int *)malloc(arr1.size*sizeof(int));
    arr1.length=0;
```

```c
    do
    {
    printf("\n\nMenu\n");
    printf("1. Insert\n");
    printf("2. Delete\n");
    printf("3. Search\n");
    printf("4. Sum\n");
    printf("5. Display\n");
    printf("6.Exit\n");

    printf("enter you choice ");
    scanf("%d",&ch);

    switch(ch)
    {
        case 1: printf("Enter an element and index
");
            scanf("%d%d",&x,&index);
            Insert(&arr1,index,x);
            break;
        case 2: printf("Enter index ");
            scanf("%d",&index);
            x=Delete(&arr1,index);
            printf("Deleted Element is %d\n",x);
            break;
        case 3:printf("Enter element to search ");
            scanf("%d",&x);
            index=LinearSearch(&arr1,x);
            printf("Element index %d",index);
            break;
        case 4:printf("Sum is %d\n",Sum(arr1));
            break;
        case 5:Display(arr1);

    }
    }while(ch<6);
    return 0;
}
```