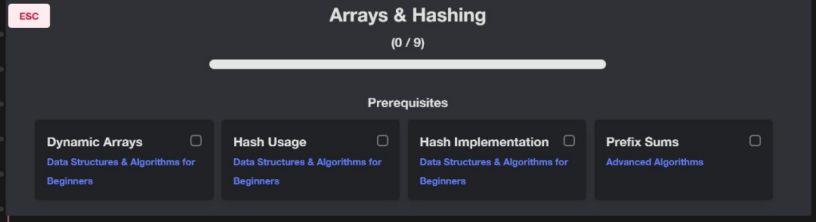



```
#include <bits/stdc++.h>
using namespace std;
// Insert n into arr at the next open position
// length is the number of 'real' value in arr, and capacity is the size
// (aka memory allocated for the fixed size array).
void insertEnd(int arr[], int n, int length, capacity){
    if( length < capacity){
        arr[length] = n;
    }
}

// Remove from the last position in the array if the array is not empty
// (ie. length is non-zero).
void removeEnd( int arr[], int length){
    if( length > 0){
        arr[length-1] = 0;
    }
}
// Insert n into index i after shifting elements to the right.
// Assuming i is a valid index and arr is not full.
void insertMiddle( int arr[], int i, int n, int length){
    for( int index = length-1; index >= i; index--){
        arr[index+1] = arr[index];
    }
    arr[i] = n;
}
```

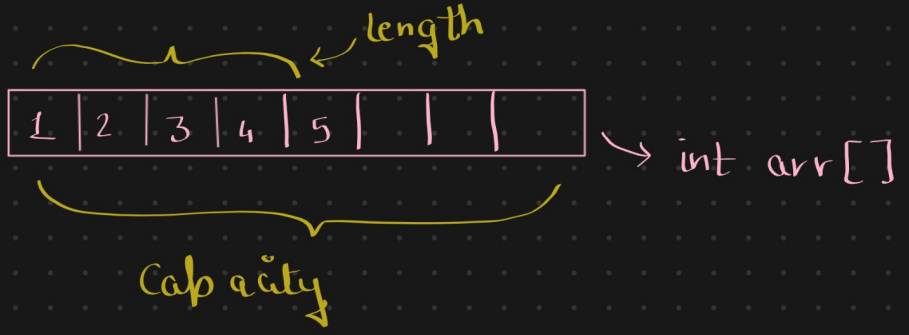
```
// Remove value at index i before shifting to the left.
// Assuming i is a valid index.
void removeMiddle(int arr[], int i, int length){
    for(int index= i+1; index< length; index++){
        arr[index-1] = arr[index];
    }
}
void printArr(int arr[], int capacity){
    for(int i=0; i< capacity; i++){
        cout<<arr[i] << ' ';
    }
    cout<< endl;
}
```

→

Self Practice

1) InsertEnd 2) RemoveEnd 3) InsertMiddle 4) Remove middle 5) PrintArr

1) InsertEnd →

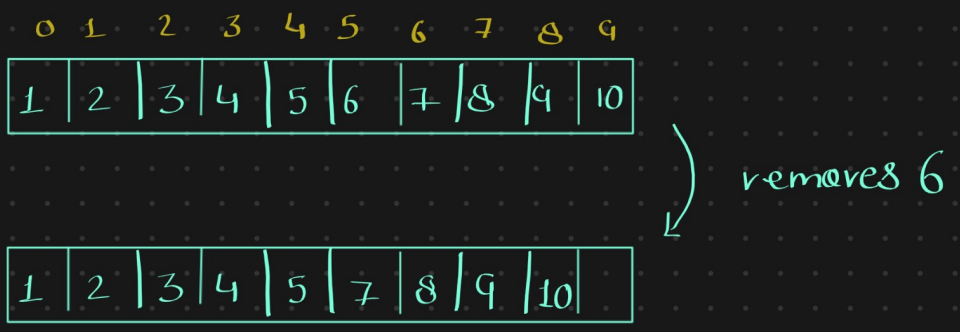



```
void insertEnd(int arr[], int n, int length, int capacity){
    if(length< capacity){
        arr[length+1] = n;
    }
```

2) RemoveEnd → just want to remove index.

```
void removeEnd(int arr[], int length)
    if( length >0) {
        arr[length-1] =0;
    }
```

3) InsertMiddle



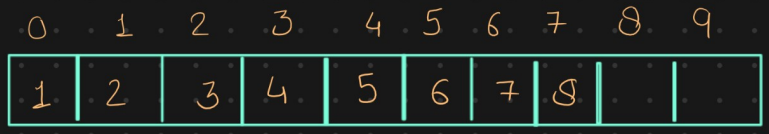
// remove the index i first; before shifting the element to the left

```
void insertMiddle(int arr[], int i , int length){
    for( int index = length-1, index >= i , index--){
        arr[index-1] = arr[index];
    }
    arr[i]
```

4) InsertMiddle

↳

// Insert n into index i after shifting it to the right




```
void insertMiddle(int arr[],int
```

→

Practice Set → 2/4/25

1) removeEnd

```
void removeEnd(int arr[], int length){
    if(length > 0){
        arr[length-1] = 0;
    }
```

2) insertEnd(int arr[], int n, int length, int capacity){

```
    if(length < capacity){
        arr[length] = n;
    }
```

3) removeMiddle

```
void removeMiddle(int arr[], int i, int length){
    for(int index = length-1; index >= i; index--){
        arr[index-1] = arr[index];
    }
}
```

4) insertMiddle


```
void insertMiddle(int arr[], int n, int length, int i){
    for(int index = i; index < length; i++){
        arr[i+1] = arr[i];
    }
    arr[i] = n;
}
```

5)
```
void printArr(int arr[]; int capacity){
    for(int i=0; i < capacity; i++){
        cout << arr[i] << ' ';
    }
}
```

→ Practice day → 8th April, 2025

1) InsertEnd 2) RemoveEnd 3) InsertMiddle 4) RemoveMiddle
5) printArr

```
#include<bits/stdc++.h>
using namespace std;

void removeEnd(int arr[], int length){
    if(length>0){
        arr[length]=0;
    }
}

void insertEnd(int arr[], int n, int length, int capacity){
    if(length<capacity){
        arr[length-1] = n;
    }
}

void removeMiddle(int arr[], int i, int length){
// assuming the length is valid
        arr[i]=0;
        for(int index=length-1; index>=length; index--){
            arr[index-1]= arr[index];
        }
}
```

```
void insertMiddle(int arr[], int i, int n, int length){
// assuming length is valid.
    for(int index = i, index < length, index++){
        arr[index+1] = arr[index];
    }
    arr[i] = n;
}

void printArr(int arr[], int length){
    for(int index=0; index < length; index++){
        cout << arr[index] << ' ';
    }
}
```

practice day 12/4/25

1) insertEnd 2) removeEnd 3) insertMiddle 4) insertMiddle
5) printArr

→

```
# include <bits/stdc++.h>
using namespace std;

void insertEnd(int arr[], int n, int length, int capacity){
    if (length < capacity){
        arr[length] = n;
    }
}
void removeEnd(int arr[], int length){
    if (length > 0){
        arr[length - 1] = 0;
    }
}
void insertMiddle(int arr[]; int length, int n, int index){
    for(int index = i; i < length; i++){
        arr[i+1] = arr[i];
    }
    arr[i] = n;
}

void removeMiddle(int arr[]; int i, int length){
    for(int index = length-1; index >= i, index--){
        arr[index-1] = arr[index];
```

```
    }
}
void printArray(int arr[]; int length){
    for(int i=0; i< length; i++){
        cout << arr[i] << endl;
    }
}
```