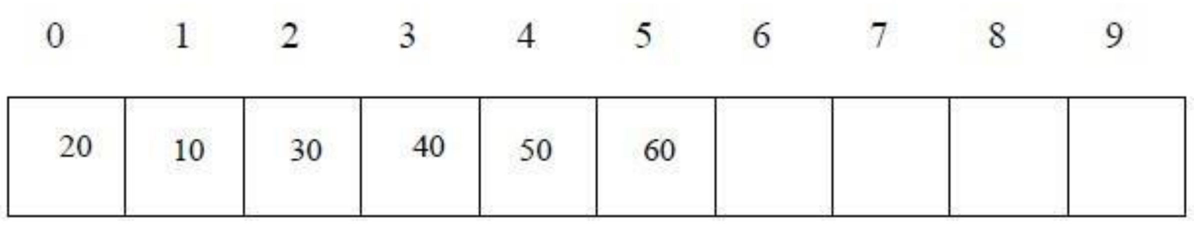
2.4 ARRAY IMPLEMENTATION OF LIST ADT

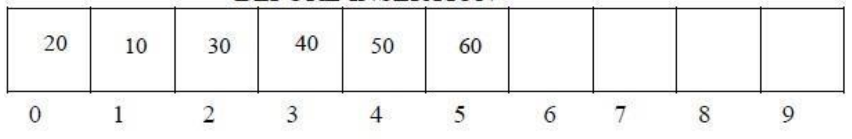
- Array is a collection of data stored in a consecutive memory location.
- Insertion and Deletion operation are expensive as it requires more data movement



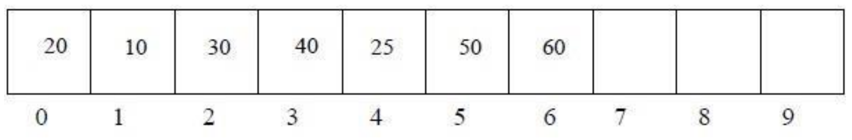
Insertion

- Insertion refers to the operation of adding(storing) an element to the list at the specified position.
- When we insert an element into a position, the elements from that particular position to the last element are move forward one position.

BEFORE INSERTION



AFTER INSERTING 25 IN THE POSITION 4

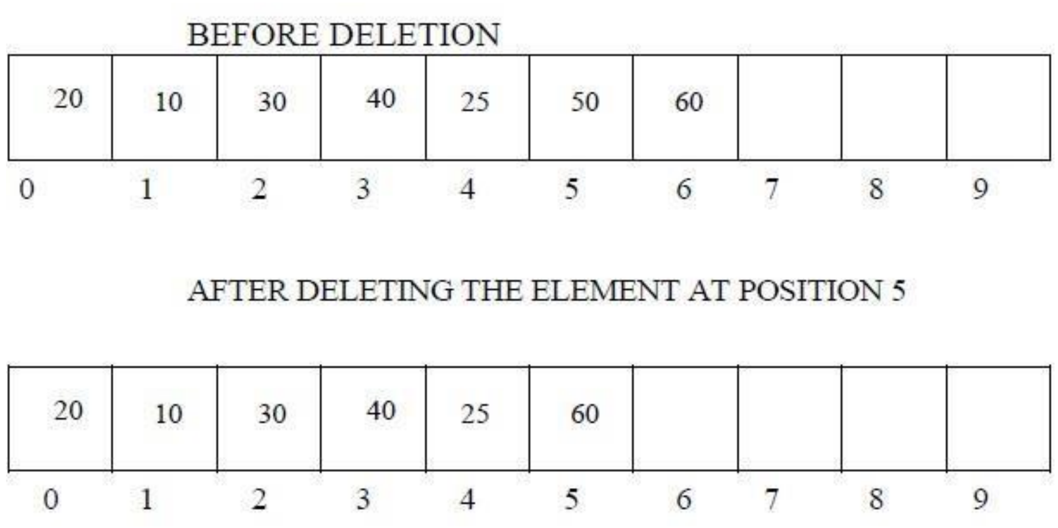


<u>C++ code for Insert( ):</u>

```
void insertElement(int pos, int item)
{
    if (size == MAX)
    {
        cout << "\nList is Full";
        return;
    }

     if (pos < 1 || pos > size + 1)
    {
        cout << "\nInvalid Position";
        return;
    }

    for (int i = size; i >= pos; i--)
        arr[i] = arr[i - 1];

    arr[pos - 1] = item;
    size++;

    cout << "\nElement Inserted";
}
```

Deletion

- Deletion refers to the operation of removing an element from the list at the specified position.
- When we delete an element from a position, the elements after that position are move backward one position.



C++ code for delete( ):

```
void deleteElement(int pos)
    {
        if (size == 0)
        {
            cout << "\nList is Empty";
            return;
        }

        if (pos < 1 || pos > size)
        {
            cout << "\nInvalid Position";
            return;
        }

        for (int i = pos - 1; i < size - 1; i++)
            arr[i] = arr[i + 1];

        size--;
        cout << "\nElement Deleted";
    }
```

Searching

Searching is a process of finding the position of an element.

C++ code for search( ):

```
void search(int Element)
    {
        for (int i = 0; i < size; i++)
        {
            if (arr[i] == Element)
            {
                cout << "\nElement found at position: " << i + 1;
                return;
            }
        }
        cout << "\nElement not found";
    }
```

Traverse or Display

Traverse means print the elements in the list

C++ code for display( ):

```
void display()
    {
        if (size == 0)
        {
            cout << "\nList is Empty";
            return;
        }
        cout << "\nList Elements: ";
        for (int i = 0; i < size; i++)
            cout << arr[i] << " ";
    }
```

Advantages

- Simple and easy to implement
- Fast access using index
- Efficient for small lists

Drawbacks in arrays:

- Has a fixed size.
- Insertion and deletion are costly due to shifting.
- Memory may be wasted if the array is not fully utilized.