**4.1 Consider the linear arrays AAA(5: 50), BBB(-5: 10) and CCC(18).**

**(a) Find the number of elements in each array.**

**(b) Suppose Base(AAA) = 300 and w = 4 words per memory cell for AAA. Find the**

**address of AAA[15]. AAA[35] and AAA[55].**

**Ans**

**(a)**

**The number of elements is equal to the length; hence use the formula**

**Length = UB - LB + 1**

**Accordingly, Length(AAA) = 50 - 5 + 1 = 46**

**Length(BBB) = 10 - (-5) + 1 = 16**

**Length(CCC) = 18 - l + l = 18**

**Note that Length(CCC) = UB, since LB = 1.**

**(b)**

**Use the formula**

**LOC(AAA[K]) = Base(AAA) + w{K - LB)**

**LOC(AAA[15])= 300 + 4(15 5) = 340**

**LOC(AAA[35)) = 300 + 4(35 5) = 420**

**Hence: AAA[55] is not an element of AAA, since 55 exceeds UB = 50.**

**4.2 Suppose a company keeps a linear array YEAR(1920: 1970) such that YEAR[K] contains the number of employees born in year K. Write a module for each of the following tasks:**

**(a) To print each of the years in which no employee was born.**

**(b) To find the number NNN of years in which no employee was born.**

**(c) To find the number N50 of employees who will be at least 50 years old at the end of**

**the year. (Assume 1984 is the current year.)**

**(d) To find the number NL of employees who will be at least L years old at the end of the**

**year. (Assume 1984 is the current year.)**

**Ans**

**Each module traverses the array.**

**(a)**

**1. Repeat for K = 1920 to 1970:**

**If YEAR[K] = 0, then: Write: K.**

**[End of loop.]**

**2. Return.**

**(b)**

**1. Set NNN := 0.**

**2. Repeat for K = 1920 to 1970:**

**If YEAR[K] = 0, then: Set NNN = NNN+ 1.**

**[End of loop.]**

**3. Return.**

**(c)**

**We want the number of employees born in 1934 or earlier.**

**1. Set N50 := 0.**

**2. Repeat for K = 1920 to 1934:**

**Set N50 = N50 + YEAR[K]. [End of loop.]**

**3. Return.**

**(d)**

**We want the number of employees born in year 1984 - L or earlier.**

**1. Set NL = 0 and LLL := 1984-L**

**2. Repeat for K = 1920 to LLL:**

**Set NL := NL + YEAR[K].**

**[End of loop.]**

**3. Return.**

**4.3 Suppose a 10-element array A contains the values a, a2, ..., a10. Find the values in A after each loop.**

**(a)**

**Repeat for K = I to 9:**

**Set A[K + 1] := A[K].**

**[End of loop.]**

**(b)**

**Repeat for K =9 to 1 by -1:**

**Set A[K + 1] = A[k].**

**[End of loop.]**

**Note that the index K runs from I to 9 in part (a) but in reverse order from 9 back to 1 in part (b).**

**Ans**

**(a)**

**First A[2] = A[I] sets A[2] = a1, the value of A[l].**

**Then A[3] = A[2] sets A[3] = a1, the current value of A[2].**

**Then A[4] = A[3] sets A[4| = a1, the current value of A[3].**

**And so on. Thus every element of A will have the value a1, the original value of A[1].**

**(b)**

**First A[10]: = A[9] sets A[10] = a9.**

**Then A[9]:= A[8] sets A[9] = a8**

**Then A[8]: = A[7] sets A[8] = a7. And so on.**

**Thus every value in A will move to the next location. At the end of the loop, we still have A[l] = a1**

**4.4 Consider the alphabetized linear array NAME in Fig.**

|  |  |
| --- | --- |
|  | **NAME** |
| **1** | **Allen** |
| **2** | **Clark** |
| **3** | **Dickens** |
| **4** | **Edwards** |
| **5** | **Goodman** |
| **6** | **Hobbs** |
| **7** | **Irwin** |
| **8** | **Klein** |
| **9** | **Lewis** |
| **10** | **Morgan** |
| **11** | **Richards** |
| **12** | **Scott** |
| **13** | **Tucker** |
| **14** | **Walton** |

**(a) Find the number of elements that must be moved if Brown, Johnson and Peters are**

**inserted into NAME at three different times.**

**(b) How many elements are moved if the three names are inserted at the same time?**

**(c) How does the telephone company handle insertions in a telephone directory?**

**Ans**

**(a)**

**Inserting Brown requires 13 elements to be moved, inserting Johnson requires 7 elements to be moved and inserting Peters requires 4 elements to be moved. Hence 24 elements are moved.**

**(b)**

**If the elements are inserted at the same time, then 13 elements need be moved, each only once.**

**(c)**

**The telephone company keeps a running list of new numbers and then updates the**

**telephone directory once a year.**

**4.5 Consider the alphabetized linear array NAME in above Fig.**

**(a) Using the linear search algorithm, how many comparisons C are used to**

**locate Hobbs, Morgan and Fisher?**

**(b) Indicate how the algorithm may be changed for such a sorted array to make an**

**unsuccessful search more efficient. How does this affect part (a).**

**Ans**

**(a)**

**C(Hobbs) = 6, since Hobbs is compared with each name, beginning with Allen, until**

**Hobbs is found in NAME[6].**

**C(Morgan) = 10. since Morgan appears in NAME[10|.**

**C(Fisher) = 15, since Fisher is initially placed in NAME[15] and then Fisher is compared**

**with every name until it is found in NAME[15). Hence the search is unsuccessful.**

**(b)**

**Observe that NAME is alphabetized. Accordingly, the linear search can stop after a**

**given name XXX is compared with a name YYY such that XXX < YYY (i.e., such that,**

**alphabetically, XXX comes before YYY). With this algorithm, C(Fisher) = 5, since the**

**search can stop after Fisher is compared with Goodman in NAME[5].**

**4.6 Suppose the binary search algorithm, is applied to the array NAME in Fig to find the location of Goodman. Find the ends BEG and END and the middle MID for the test segment in each step of the algorithm.**

**Ans**

**Recall that MID = INT(BEG + END)/2), where INT means integer value.**

**Step 1. Here BEG = 1 [Allen] and END = 14 [Walton), so MID = 7 [Irwin).**

**Step 2. Since Goodman < Irwin, reset END = 6. Hence MID = 3 [Dickens].**

**Step 3. Since Goodman> Dickens, reset BEG = 4. Hence MID = 5 [Goodman].**

**We have found the location LOC = 5 of Goodman in the array. Observe that, there were**

**C=3 comparisons.**

**4.7 Modify the binary search algorithm, so that it becomes a search and insertion**

**algorithm.**

**Ans**

**There is no change in the first four steps of the algorithm given below.**

1. **[Initialize segment variables.]**

**Set BEG:= LB, END := UB and MID = INT(BEG + END)/2).**

1. **Repeat Steps 3 and 4 while BEG < = END and DATA[MID] != ITEM.**
2. **If ITEM < DATA[MID], then:**

**Set END:= MID - 1.**

**Else:**

**Set BEG := MID + 1.**

**[End of If structure.]**

1. **Set MID:= INT(BEG + END)/2).**

**[End of Step 2 loop]**

**The algorithm transfers control to Step 5 only when ITEM does not appear in DATA. In such a case, ITEM is inserted before or after DATA[MID] according to whether ITEM < DATA[MID] or ITEM > DATA[MID]. The algorithm follows.**

**Binary Search and Insertion**

**DATA is a sorted array with N elements, and ITEM is a given item of information. This algorithm finds the location LOC of ITEM in DATA or inserts ITEM in its proper place in DATA. Steps 1 through 4. Same as in Binary Search Algorithm written above.**

**5. If ITEM < DATA[MID), then:**

**Set LOC:= MID.**

**Else:**

**Set LOC := MID + 1.**

**[End of If structure.]**

**6. Insert ITEM into DATA[LOC] using Insertion Algorithm.**

**7. Exit.**

**4.8 Using the bubble sort algorithm, find the number C of comparisons and the**

**number D of interchanges which alphabetize the n = 6 letters in PEOPLE.**

**Ans**

**The sequences of pairs of letters which are compared in each of the n - 1 =5 passes follow:**

**a square indicates that the pair of letters is compared and interchanged, and a circle indicates that the pair of letters is compared but not interchanged.**

****

**Since n = 6, the number of comparisons will be C = 5 +4 +3 +2+ 1 = 15. The number D**

**of interchanges depends also on the data, as well as on the number n of elements. In this**

**case D =9.**