

Data Structures and Algorithms Lab Manual

Contents

1 Bubble Sort

Problem Statement

Design, Develop and Implement a program in C to sort a given set of N integer elements using Bubble Sort technique.

Source Code

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void read(int b[100], int n)
5 {
6     int i;
7     for(i=0; i<n; i++)
8         scanf("%d", &b[i]);
9 }
10
11 void display(int b[100], int n)
12 {
13     int i;
14     for(i=0; i<n; i++)
15         printf("%d\t", b[i]);
16 }
17
18 void sort(int b[100], int n)
19 {
20     int i, j, t, swap=0, comp=0, flag=0;
21     for(i=0; i<n-1; i++)
22     {
23         for(j=0; j<n-i-1; j++)
24         {
25             if(b[j] > b[j+1])
26             {
27                 t = b[j];
28                 b[j] = b[j+1];
29                 b[j+1] = t;
30                 swap++;
31                 flag=1;
32             }
33             comp++;
34         }
35         if(flag==0)
36         {
37             printf("\nNo of swaps: %d\nNo of comparisons: %d",
38                 swap, comp);
39             return;
40         }
41     }
```

```

41     printf("\nNo of swaps: %d\nNo of comparisons: %d", swap, comp
42         );
43 }
44 int main()
45 {
46     int b[100], n;
47     printf("Enter number of elements: ");
48     scanf("%d", &n);
49     printf("Enter the elements: ");
50     read(b, n);
51     printf("Elements are: ");
52     display(b, n);
53     sort(b, n);
54     printf("\nSorted array: ");
55     display(b, n);
56     return 0;
57 }

```

Listing 1: Bubble Sort Implementation

Output

```

Enter the number of elements: 5
Enter the elements: 50 40 30 20 10
Sorted array: 10 20 30 40 50

```

2 Selection Sort

Problem Statement

Design, Develop and Implement a program in C to sort a given set of N integer elements using Selection Sort technique.

Source Code

```
1 #include <stdio.h>
2
3 void selection_sort(int a[], int n) {
4     int i, j, min, temp;
5     for(i=0; i<n-1; i++) {
6         min = i;
7         for(j=i+1; j<n; j++) {
8             if(a[j] < a[min])
9                 min = j;
10        }
11        temp = a[i];
12        a[i] = a[min];
13        a[min] = temp;
14    }
15 }
16
17 int main() {
18     int a[100], n, i;
19     printf("Enter the number of elements: ");
20     scanf("%d", &n);
21     printf("Enter the elements: ");
22     for(i=0; i<n; i++)
23         scanf("%d", &a[i]);
24
25     selection_sort(a, n);
26
27     printf("\nSorted array: ");
28     for(i=0; i<n; i++)
29         printf("%d\t", a[i]);
30     printf("\n");
31     return 0;
32 }
```

Listing 2: Selection Sort Implementation

Output

```
Enter the number of elements: 5
Enter the elements: 30 50 10 20 40
Sorted array: 10 20 30 40 50
```

3 Insertion Sort

Problem Statement

Design, Develop and Implement a program in C to sort a given set of N integer elements using Insertion Sort technique.

Source Code

```
1 #include <stdio.h>
2
3 void insertion_sort(int a[], int n) {
4     int i, j, item;
5     for(i=1; i<n; i++) {
6         item = a[i];
7         j = i-1;
8         while(j>=0 && a[j] > item) {
9             a[j+1] = a[j];
10            j--;
11        }
12        a[j+1] = item;
13    }
14 }
15
16 int main() {
17     int a[100], n, i;
18     printf("Enter the number of elements: ");
19     scanf("%d", &n);
20     printf("Enter the elements: ");
21     for(i=0; i<n; i++)
22         scanf("%d", &a[i]);
23
24     insertion_sort(a, n);
25
26     printf("\nSorted array: ");
27     for(i=0; i<n; i++)
28         printf("%d\t", a[i]);
29     printf("\n");
30     return 0;
31 }
```

Listing 3: Insertion Sort Implementation

Output

```
Enter the number of elements: 5
Enter the elements: 5 4 3 2 1
Sorted array: 1 2 3 4 5
```

4 Linear Search

Problem Statement

Design, Develop and Implement a program in C to search a key element in a given set of N integer elements using Linear Search.

Source Code

```
1 #include <stdio.h>
2
3 int linear_search(int a[], int n, int key) {
4     int i;
5     for(i=0; i<n; i++) {
6         if(a[i] == key)
7             return i;
8     }
9     return -1;
10 }
11
12 int main() {
13     int a[100], n, i, key, pos;
14     printf("Enter the number of elements: ");
15     scanf("%d", &n);
16     printf("Enter the elements: ");
17     for(i=0; i<n; i++)
18         scanf("%d", &a[i]);
19
20     printf("Enter the key element to search: ");
21     scanf("%d", &key);
22
23     pos = linear_search(a, n, key);
24
25     if(pos == -1)
26         printf("Element not found\n");
27     else
28         printf("Element found at position %d\n", pos+1);
29
30     return 0;
31 }
```

Listing 4: Linear Search Implementation

Output

```
Enter the number of elements: 5
Enter the elements: 10 20 30 40 50
Enter the key element to search: 30
Element found at position 3
```

5 Binary Search

Problem Statement

Design, Develop and Implement a program in C to search a key element in a given set of N integer elements using Binary Search.

Source Code

```
1 #include <stdio.h>
2
3 int binary_search(int a[], int n, int key) {
4     int low = 0, high = n-1, mid;
5     while(low <= high) {
6         mid = (low + high) / 2;
7         if(a[mid] == key)
8             return mid;
9         else if(a[mid] < key)
10            low = mid + 1;
11        else
12            high = mid - 1;
13    }
14    return -1;
15 }
16
17 int main() {
18     int a[100], n, i, key, pos;
19     printf("Enter the number of elements: ");
20     scanf("%d", &n);
21     printf("Enter the elements (in sorted order): ");
22     for(i=0; i<n; i++)
23         scanf("%d", &a[i]);
24
25     printf("Enter the key element to search: ");
26     scanf("%d", &key);
27
28     pos = binary_search(a, n, key);
29
30     if(pos == -1)
31         printf("Element not found\n");
32     else
33         printf("Element found at position %d\n", pos+1);
34
35     return 0;
36 }
```

Listing 5: Binary Search Implementation

Output

Enter the number of elements: 5

Enter the elements (in sorted order): 10 20 30 40 50

Enter the key element to search: 40

Element found at position 4

6 String Matching

Problem Statement

Design, Develop and Implement a program in C to find the position of a pattern string in a text string using Naive String Matching algorithm.

Source Code

```
1 #include <stdio.h>
2 #include <string.h>
3
4 int string_match(char t[], char p[]) {
5     int i, j, m, n;
6     n = strlen(t);
7     m = strlen(p);
8     for(i=0; i <= n-m; i++) {
9         j = 0;
10        while(j < m && p[j] == t[i+j]) {
11            j++;
12        }
13        if(j == m)
14            return i;
15    }
16    return -1;
17 }
18
19 int main() {
20     char t[100], p[50];
21     int pos;
22
23     printf("Enter the text: ");
24     scanf("%s", t);
25     printf("Enter the pattern: ");
26     scanf("%s", p);
27
28     pos = string_match(t, p);
29
30     if(pos == -1)
31         printf("Pattern not found\n");
32     else
33         printf("Pattern found at position %d\n", pos+1);
34
35     return 0;
36 }
```

Listing 6: String Matching Implementation

Output

Enter the text: Hello_World

Enter the pattern: World
Pattern found at position 7

7 Merge Sort

Problem Statement

Design, Develop and Implement a program in C using divide and conquer technique for Merge Sort.

Source Code

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int a[100], c[100];
5
6 void read(int n)
7 {
8     int i;
9     printf("Enter the elements: ");
10    for(i=0; i<n; i++)
11        scanf("%d", &a[i]);
12 }
13
14 void display(int n)
15 {
16     int i;
17     for(i=0; i<n; i++)
18         printf("%d\t", a[i]);
19 }
20
21 void merge(int low, int mid, int high)
22 {
23     int i, j, k;
24     i = low;
25     j = mid + 1;
26     k = low;
27
28     while(i <= mid && j <= high)
29     {
30         if(a[i] <= a[j])
31             c[k] = a[i++];
32         else
33             c[k] = a[j++];
34         k++;
35     }
36
37     while(i <= mid)
38     {
39         c[k] = a[i];
40         i++;
41         k++;
42     }
```

```

43
44     while(j <= high)
45     {
46         c[k] = a[j];
47         j++;
48         k++;
49     }
50
51     for(i=low; i<=high; i++)
52         a[i] = c[i];
53 }
54
55 void mergesort(int low, int high)
56 {
57     if(low < high)
58     {
59         int mid = (low + high) / 2;
60         mergesort(low, mid);
61         mergesort(mid + 1, high);
62         merge(low, mid, high);
63     }
64 }
65
66 int main()
67 {
68     int n;
69     printf("Enter the number of elements: ");
70     scanf("%d", &n);
71     read(n);
72     printf("Elements are: ");
73     display(n);
74     mergesort(0, n-1);
75     printf("\nSorted array: ");
76     display(n);
77     return 0;
78 }

```

Listing 7: Merge Sort Implementation

Output

```

Enter the number of elements: 5
Enter the elements: 10 30 50 20 40
Elements are: 10 30 50 20 40
Sorted array: 10 20 30 40 50

```

Appendix: Quick Sort (Extra)

Problem Statement

Design, Develop and Implement a program in C using divide and conquer technique for Quick Sort.

Source Code

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int a[100];
5
6 void read(int n)
7 {
8     int i;
9     printf("Enter the elements: ");
10    for(i=0; i<n; i++)
11        scanf("%d", &a[i]);
12 }
13
14 void display(int n)
15 {
16     int i;
17     for(i=0; i<n; i++)
18         printf("%d\t", a[i]);
19 }
20
21 int partition(int low, int high)
22 {
23     int p = a[low], i = low, j = high + 1, temp;
24     do
25     {
26         do
27         {
28             i++;
29         } while (p >= a[i] && i < high);
30
31         do
32         {
33             j--;
34         } while (p < a[j]);
35
36         if (i < j)
37         {
38             temp = a[i];
39             a[i] = a[j];
40             a[j] = temp;
41         }
42     } while (i < j);
```

```

43
44     temp = a[j];
45     a[j] = a[low];
46     a[low] = temp;
47     return j;
48 }
49
50 void quicksort(int low, int high)
51 {
52     if (low < high)
53     {
54         int k = partition(low, high);
55         quicksort(low, k - 1);
56         quicksort(k + 1, high);
57     }
58 }
59
60 int main()
61 {
62     int n;
63     printf("Enter the number of elements: ");
64     scanf("%d", &n);
65     read(n);
66     printf("Elements are: ");
67     display(n);
68     quicksort(0, n - 1);
69     printf("\nSorted array: ");
70     display(n);
71     return 0;
72 }

```

Listing 8: Quick Sort Implementation