Programming Lab

Assignment – 4

Question:-

1. Write a menu-driven program to implement the following searching techniques using an array (a) Linear or sequential search (b) Binary search

2. Write a menu-driven program to implement the following sorting techniques using an array

(a) Bubble sort (b) Insertion sort (c) Selection sort

3. Write a menu-driven program to implement the following sorting techniques using a single linked list (interchanging data)

(a) Bubble sort (b) Insertion sort (c) Selection sort

4. Write a menu-driven program to implement the following sorting techniques using a single linked list (interchanging links)

(a) Bubble sort (b) Insertion sort (c) Selection sort

5. Write a menu-driven program to implement the following sorting techniques using an array (recursive functions)

(a) Quick sort (b) Merge sort

Program 1:-

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

void linear_search(int a[],int n,int val){
    int flag = 0,i;
    for(i=0;i<n;i++){
        if(a[i] == val){
            printf("%d found at index %d\n",val,i);
            flag = 1;
        }
    }
    if(flag == 0){
        printf("Element not found\n");
    }
}

void binary_search(int a[],int n,int val){
    int left = 0;
    int right = n-1;
    int flag = 0;
    while(left<right){
        int mid = (left + right) / 2;
        if(a[mid] == val){
            printf("%d found at index %d\n",val,mid);
            flag = 1;
            break;
        }
        else if(val < a[mid]){
            right = mid - 1;
        } else{
            left = mid + 1;
        }
    }
    if(flag == 0){
        printf("Element not found\n");
    }
}

int main(){
    int op,val,n;
    printf("Enter the size of the array\n");
    scanf("%d",&n);
    int arr[n];
    printf("Enter the values\n");
    for(int i=0;i<n;i++){
```

```
        scanf("%d",&arr[i]);
    }
    printf("1 for linear search\n2 for binary search\n");
    while (1){
        printf("Enter your operation\n");
        scanf("%d",&op);
        switch (op){
            case 1:
                printf("Enter the value you want to search...\n");
                scanf("%d",&val);
                linear_search(arr,n,val);
                break;
            case 2:
                printf("Enter the value you want to search...\n");
                scanf("%d",&val);
                binary_search(arr,n,val);
                break;
            default:
                exit(0);
        }
    }

}
```

Program 2:-

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

void swap(int *a,int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}

void bubble_sort(int a[],int n){
    int flag = 0;
    for(int i=0;i<n;i++){
        for(int j=0;j<n-1-i;j++){
            if(a[j]>a[j+1]){
                swap(&a[j],&a[j+1]);
                flag = 1;
            }
        }
        if(flag == 0){
            break;
        }
```

```
    }
    printf("Array Sorted\n");
}

void insertion_sort(int arr[],int n){
    for(int i=1;i<n;i++){
        int val = arr[i];
        int hole = i;
        while(hole > 0 && arr[hole-1] > val){
            arr[hole] = arr[hole - 1];
            hole--;
        }
        arr[hole] = val;
    }
    printf("Array sorted...\n");
}

void selection_sort(int a[],int n){
    for(int i=0;i<n;i++){
        int min = i;
        for(int j=i+1;j<n;j++){
            if(a[j] < a[min]){
                min= j;
            }
        }
        swap(&a[i],&a[min]);
    }
    printf("Array sorted...\n");
}

void display(int a[],int n){
    for(int i=0;i<n;i++){
        printf("%d ",a[i]);
    }
    printf("\n");
}

int main(){
    int op,n;
    printf("Enter the size of the array\n");
    scanf("%d",&n);
    int arr[n];
    printf("Enter the values\n");
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    printf("1 for bubble sort\n2 for insertion sort\n3 for selection sort\n4 f
or display\n");
```

```
    while (1){
        printf("Enter your operation\n");
        scanf("%d",&op);
        switch (op){
            case 1:
                bubble_sort(arr,n);
                break;
            case 2:
                insertion_sort(arr,n);
                break;
            case 3:
                selection_sort(arr,n);
                break;
            case 4:
                printf("Elments of the array are...\n");
                display(arr,n);
                break;
            default:
                exit(0);
        }
    }
}
```

Program 3:-

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

struct node{
    int data;
    struct node *next;
};
struct node *head=NULL;
int size;

void create(){
    int n,x;
    struct node *temp;
    printf("Enter the number of node you want to insert...\n");
    scanf("%d",&n);
    size = n;
    printf("Enter the values...\n");
    for(int i=0;i<n;i++){
        scanf("%d",&x);
        struct node *newnode = (struct node *)malloc(sizeof(struct node));
        newnode->data = x;
        newnode->next = NULL;
```

```
        if(head == NULL){
            head = newnode;
        }
        else{
            temp = head;
            while(temp->next){
                temp = temp->next;
            }
            temp->next = newnode;
        }
    }
}

void swap(int a,int b){
    struct node *temp=head,*temp1=NULL,*temp2=NULL;
    while(temp){
        if(temp->data == a || temp->data == b){
            if(temp1== NULL)
                temp1 = temp;
            else
                temp2 = temp;
        }
        temp = temp->next;
    }
    int tmp = temp1->data;
    temp1->data = temp2->data;
    temp2->data = tmp;
}

void bubble_sort(){
    struct node *temp1,*temp2;
    int flag = 0;
    for(int i=0;i<size;i++){
        temp1 = head;
        temp2 = temp1->next;
        for(int j=0;j<size - 1 - i;j++){
            if(temp1->data > temp2->data){
                swap(temp1->data,temp2->data);
                flag = 1;
            }
            temp1 = temp1->next;
            temp2 = temp2->next;
        }
        if(flag==0){
            break;
        }
    }
    printf("List sorted...\n");
```

```
}

struct node *sortedInsert(struct node *sorted,struct node *curr){
    if(sorted == NULL || sorted->data > curr->data){
        curr->next = sorted;
        sorted = curr;
    }else{
        struct node *temp = sorted;
        while(temp->next != NULL && temp->next->data <curr->data){
            temp = temp->next;
        }
        curr->next = temp->next;
        temp->next = curr;
    }
    return sorted;
}

void insertion_sort(){
    struct node *sorted = NULL,*curr = head;
    while(curr){
        struct node *next = curr->next;
        sorted = sortedInsert(sorted,curr);
        curr = next;
    }
    head= sorted;
    printf("List sorted...\n");
}

void selection_sort(){
    struct node *ptr1 = head,*ptr2;
    int min;
    while(ptr1){
        ptr2 = ptr1->next;
        min = ptr1->data;
        while(ptr2){
            if(ptr2->data < min){
                min = ptr2->data;
            }
            ptr2 = ptr2->next;
        }
        if(min != ptr1->data)
            swap(ptr1->data,min);
        ptr1 = ptr1->next;
    }
    printf("List sorted...\n");
}

void display(){
```

```c
    struct node *temp = head;
    while(temp){
        printf("->%d",temp->data);
        temp=temp->next;
    }
    printf("\n");
}

int main(){
    int op,a,b;
    printf("1 to create the list\n2 to display the list\n3 to bubble sort\n4 f
or insertion sort\n5 for selection sort\n");
    while(1){
        printf("Enter your operation\n");
        scanf("%d",&op);
        switch (op)
        {
        case 1:
            head = NULL;
            create();
            break;
        case 2:
            display();
            break;
        case 3:
            bubble_sort();
            break;
        case 4:
            insertion_sort();
            break;
        case 5:
            selection_sort();
            break;
        default:
            exit(0);
        }
    }
}
```

Program 4:-

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

struct node{
    int data;
    struct node *next;
};
struct node *head=NULL;
int size;

void create(){
    int n,x;
    struct node *temp;
    printf("Enter the number of node you want to insert...\n");
    scanf("%d",&n);
    size = n;
    printf("Enter the values...\n");
    for(int i=0;i<n;i++){
        scanf("%d",&x);
        struct node *newnode = (struct node *)malloc(sizeof(struct node));
        newnode->data = x;
        newnode->next = NULL;
        if(head == NULL){
            head = newnode;
        }
        else{
            temp = head;
            while(temp->next){
                temp = temp->next;
            }
            temp->next = newnode;
        }
    }
}

void swapAddr(struct node **ptr1,struct node **ptr2){
    struct node *temp;
    temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}

void display(){
    struct node *temp = head;
    while(temp){
```

```
        printf("->%d",temp->data);
        temp=temp->next;
    }
    printf("\n");
}

void swap(int a,int b){
    struct node *temp=head,*prev1=NULL,*cur1=NULL,*prev2=NULL,*cur2=NULL;
    if(head->data == a || head->data == b){
        cur1 = head;
    }
    while(temp->next){
        if(temp->next->data == a || temp->next->data == b){
            if(cur1== NULL){
                prev1 = temp;
                cur1 = temp->next;
            }else{
                prev2= temp;
                cur2 = temp->next;
            }
        }
        temp = temp->next;
    }
    if(head == cur1){
        head = cur2;
    }else{
        prev1->next = cur2;
    }
    if(cur1 == prev2){
        temp = cur2->next;
        cur1->next = temp;
        cur2->next = cur1;
    }else{
        prev2->next = cur1;
        temp = cur2->next;
        cur2->next = cur1->next;
        cur1->next = temp;
    }
}

void bubble_sort(){
    struct node *temp1,*temp2;
    int flag = 0;
    for(int i=0;i<size;i++){
        temp1 = head;
        temp2 = temp1->next;
        for(int j=0;j<size - 1 - i;j++){
            if(temp1->data > temp2->data){
```

```
                swap(temp1->data,temp2->data);
                swapAddr(&temp1,&temp2);
                flag = 1;
            }
            temp1 = temp1->next;
            temp2 = temp2->next;
        }
        if(flag==0){
            break;
        }
    }
    printf("List sorted...\n");
}

struct node *sortedInsert(struct node *sorted,struct node *curr){
    if(sorted == NULL || sorted->data > curr->data){
        curr->next = sorted;
        sorted = curr;
    }else{
        struct node *temp = sorted;
        while(temp->next != NULL && temp->next->data <curr->data){
            temp = temp->next;
        }
        curr->next = temp->next;
        temp->next = curr;
    }
    return sorted;
}

void insertion_sort(){
    struct node *sorted = NULL,*curr = head;
    while(curr){
        struct node *next = curr->next;
        sorted = sortedInsert(sorted,curr);
        curr = next;
    }
    head= sorted;
    printf("List sorted...\n");
}

void selection_sort(){
    struct node *ptr1 = head,*ptr2;
    struct node *minAddr;
    while(ptr1){
        ptr2 = ptr1->next;
        minAddr = ptr1;
        while(ptr2){
            if(ptr2->data < minAddr->data){
```

```
                minAddr = ptr2;
            }
            ptr2 = ptr2->next;
        }
        if(minAddr->data != ptr1->data){
            swap(ptr1->data,minAddr->data);
            swapAddr(&ptr1,&minAddr);
        }
        ptr1 = ptr1->next;
    }
    printf("List sorted...\n");
}

int main(){
    int op,a,b;
    printf("1 to create the list\n2 to display the list\n3 to bubble sort\n4 f
or insertion sort\n5 for selection sort\n");
    while(1){
        printf("Enter your operation\n");
        scanf("%d",&op);
        switch (op)
        {
        case 1:
            head = NULL;
            create();
            break;
        case 2:
            display();
            break;
        case 3:
            bubble_sort();
            break;
        case 4:
            insertion_sort();
            break;
        case 5:
            selection_sort();
            break;
        default:
            exit(0);
        }
    }
}
```

Program 5:-

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

void swap(int *a,int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}

int partition(int arr[],int l,int r){
    int i = l-1,j;
    for(j=l;j<=r;j++){
        if(arr[j] < arr[r]){
            i++;
            swap(&arr[i],&arr[j]);
        }
    }
    swap(&arr[i+1],&arr[r]);
    return i+1;
}

void merge(int arr[],int l,int m,int r){
    int i,j,k,n1 = m - l + 1,n2 = r - m;
    int L[n1],R[n2];

    for(i=0;i<n1;i++){
        L[i] = arr[l+i];
    }
    for(j=0;j<n2;j++){
        R[j] = arr[m + 1 + j];
    }

    i = 0;
    j = 0;
    k = l;
    while(i < n1 && j < n2){
        if(L[i] <= R[j]){
            arr[k] = L[i];
            i++;
        }else{
            arr[k] = R[j];
            j++;
        }
        k++;
    }
```

```
    while(i < n1){
        arr[k] = L[i];
        i++;
        k++;
    }
    while(j < n2){
        arr[k] = R[j];
        j++;
        k++;
    }
}

void quickSort(int arr[],int l,int r){
    if(l<r){
        int q = partition(arr,l,r);
        quickSort(arr,l,q-1);
        quickSort(arr,q+1,r);
    }
}

void mergeSort(int arr[],int l,int r){
    if(l<r){
        int m = (l+r)/2;
        mergeSort(arr,l,m);
        mergeSort(arr,m+1,r);
        merge(arr,l,m,r);
    }
}

void display(int a[],int n){
    for(int i=0;i<n;i++){
        printf("%d ",a[i]);
    }
    printf("\n");
}

int main(){
    int op,n;
    printf("Enter the size of the array\n");
    scanf("%d",&n);
    int arr[n];
    printf("Enter the values\n");
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    printf("1 for quick sort\n2 for merge sort\n3 for display\n");
    while (1){
```

```
        printf("Enter your operation\n");
        scanf("%d",&op);
        switch (op){
            case 1:
                quickSort(arr,0,n-1);
                printf("Array sorted...\n");
                break;
            case 2:
                mergeSort(arr,0,n-1);
                printf("Array sorted...\n");
                break;
            case 3:
                display(arr,n);
                break;
            default:
                exit(0);
        }
    }
}
```

Output 1:-

```
Enter the size of the array
5
Enter the values
1
2
3
4
5
1 for linear search
2 for binary search
Enter your operation
1
Enter the value you want to search...
4
4 found at index 3
Enter your operation
2
Enter the value you want to search...
3
3 found at index 2
Enter your operation
6
```

Output 2:-

```
Enter the size of the array
5
Enter the values
3
1
2
5
4
1 for bubble sort
2 for insertion sort
3 for selection sort
4 for display
Enter your operation
1
Array Sorted
Enter your operation
4
Elments of the array are...
1 2 3 4 5
```

```
Enter the size of the array
5
Enter the values
3
1
2
5
4
1 for bubble sort
2 for insertion sort
3 for selection sort
4 for display
Enter your operation
2
Array sorted...
Enter your operation
4
Elments of the array are...
1 2 3 4 5
```

```
Enter the size of the array
5
Enter the values
3
1
2
5
4
1 for bubble sort
2 for insertion sort
3 for selection sort
4 for display
Enter your operation
3
Array sorted...
Enter your operation
4
Elments of the array are...
1 2 3 4 5
```

Output 3:-

```
1 to create the list
2 to display the list
3 to bubble sort
4 for insertion sort
5 for selection sort
Enter your operation
1
Enter the number of node you want to insert...
5
Enter the values...
3
1
2
5
4
Enter your operation
3
List sorted...
Enter your operation
2
->1->2->3->4->5
```

```
1 to create the list
2 to display the list
3 to bubble sort
4 for insertion sort
5 for selection sort
Enter your operation
1
Enter the number of node you want to insert...
5
Enter the values...
3
1
2
5
4
Enter your operation
4
List sorted...
Enter your operation
2
->1->2->3->4->5
```

```
1 to create the list
2 to display the list
3 to bubble sort
4 for insertion sort
5 for selection sort
Enter your operation
1
Enter the number of node you want to insert...
5
Enter the values...
5
4
3
2
1
Enter your operation
5
List sorted...
Enter your operation
2
->1->2->3->4->5
```

Output 4:-

```
1 to create the list
2 to display the list
3 to bubble sort
4 for insertion sort
5 for selection sort
Enter your operation
1
Enter the number of node you want to insert...
5
Enter the values...
3
1
2
5
4
Enter your operation
3
List sorted...
Enter your operation
2
->1->2->3->4->5
```

```
1 to create the list
2 to display the list
3 to bubble sort
4 for insertion sort
5 for selection sort
Enter your operation
1
Enter the number of node you want to insert...
5
Enter the values...
3
1
2
5
4
Enter your operation
4
List sorted...
Enter your operation
2
->1->2->3->4->5
```

```
1 to create the list
2 to display the list
3 to bubble sort
4 for insertion sort
5 for selection sort
Enter your operation
1
Enter the number of node you want to insert...
5
Enter the values...
5
4
3
2
1
Enter your operation
5
List sorted...
Enter your operation
2
->1->2->3->4->5
```

Output 5:-

```
Enter the size of the array
5
Enter the values
5
4
3
2
1
1 for quick sort
2 for merge sort
3 for display
Enter your operation
1
Array sorted...
Enter your operation
3
1 2 3 4 5
```

```
Enter the size of the array
5
Enter the values
5
4
3
2
1
1 for quick sort
2 for merge sort
3 for display
Enter your operation
2
Array sorted...
Enter your operation
3
1 2 3 4 5
```