Galgotias University

Plot No. 2, Sector – 17 A, Yamuna Expressway Greater Noida, Gautam Buddh Nagar, U.P., India

School Of Computing Science & Engineering

LAB FILE

Course Name : DATA STRUCTURE LAB

Course Code : MCAN1260

School : SCSE Year : 1st Semester : 2nd

Program : MCA Session : 2022-24

Submitted By :	**Submitted To :**
Name : Saksham kumar **Admission No. :** 22SCSE2030353	**A Unni Krishnan**

INDEX

Experiment – 1

1 Write a program to find the largest and the second largest elements from an array.

Code :

```
#include<stdio.h>
#define SIZE 10

void main(){

    int arr[SIZE],l,sl,i;

    printf("Enter the array elements : ");
    for(i=0; i<SIZE; i++){
        scanf("%d",&arr[i]);
    }

    printf("Array elements are : ");
    for(i=0; i<SIZE; i++){
        printf(" %d ",arr[i]);
    }

    l = sl=-1;

    for(i=1; i<SIZE; i++){
        if(arr[i]>l){
            sl = l;
            l = arr[i];
        }else if(arr[i]>sl && arr[i]!=l){
            sl = arr[i];
        }
    }

    printf("\nLargest element is %d \n",l);
    printf("Second Largest element is %d ",sl);


}
```

Output :

```
Enter the array elements : 1
2
3
4
5
6
7
8
9
10
Array elements are :  1  2  3  4  5  6  7  8  9  10
Largest element is 10
Second Largest element is 9
PS C:\Users\ksaks>
```

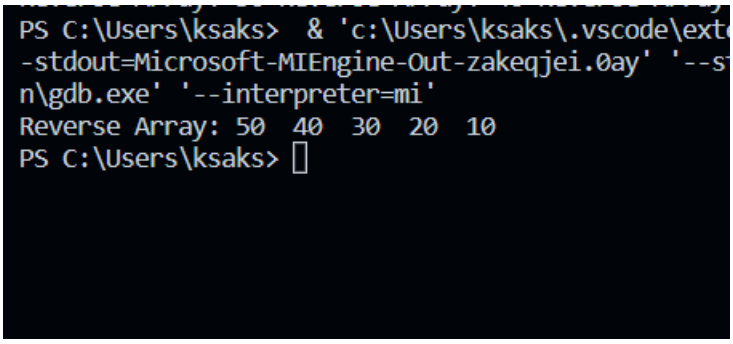
Experiment – 2

2 Write a program to reverse the contents of an array.

Code:

```
#include <stdio.h>

void main()
{
    int arr[5] = {10, 20, 30, 40, 50};
    int i = 0, j = 5;
    int c[5];
    while (i < 5)
    {
        c[i] = arr[j - 1];
        i++;
        j--;
    }

    printf("Reverse Array:");
    for (int i = 0; i < 5; i++)
    {

        printf(" %d ", c[i]);
    }
}
```

Output :



Experiment - 3

3 Write a program to merge two sorted arrays.

Code:

```
#include <stdio.h>

void main()
{
    int a[5];
    int b[7];
    int m = 5, n = 7;
    int n1 = m + n;
    int c[n1];
    int j = 0, k = 0, i = 0;

    // Insert elements in array 1
    printf("Enter %d element in the array :", m);
    for (int i = 0; i < m; i++)
    {
        scanf("%d", &a[i]);
    }
    // Insert elements in array 2
    printf("Enter %d element in the array :", n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &b[i]);
    }

    // Print elements of array 1
    printf("Array 1 elements are :");
    for (int i = 0; i < m; i++)
    {
        printf(" %d ", a[i]);
    }
    // Print elements of array 2
    printf("\nArray 2 elements are :");
    for (int i = 0; i < n; i++)
    {
        printf(" %d ", b[i]);
    }

    // Merge array1 and array2 in ascending order
    while (i < n1)
    {
        if (j < m && k < n)
        {
            if (a[j] < b[k])
```

```
            {
                c[i] = a[j];
                j++;
            }
            else
            {
                c[i] = b[k];
                k++;
            }
            i++;
        }
        else if (j == m)
        {
            while (i < n1)
            {
                c[i] = b[k];
                k++;
                i++;
            }
        }
        else
        {
            while (i < n1)
            {
                c[i] = a[j];
                j++;
                i++;
            }
        }
    }

    // Print merged ascending array
    printf("\nMerged Array elements are :");
    for (int i = 0; i < n1; i++)
    {
        printf(" %d ", c[i]);
    }
}
```

Output :

```
Enter 5 element in the array :1
2
3
4
5
Enter 7 element in the array :6
7
8
9
10
12
13
Array 1 elements are : 1  2  3  4  5
Array 2 elements are : 6  7  8  9  10  12  13
Merged Array elements are : 1  2  3  4  5  6  7  8  9  10  12  13
PS C:\Users\ksaks>
```

Experiment - 4

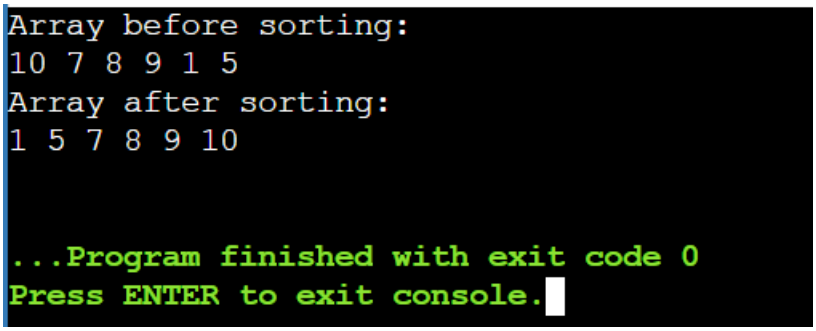
4 Write a program to implement Quick Sort.

Code :

```
#include <stdio.h>

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


int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] <= pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pivot = partition(arr, low, high);

        quickSort(arr, low, pivot - 1);
        quickSort(arr, pivot + 1, high);
    }
}

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

int main() {
    int arr[] = {10, 7, 8, 9, 1, 5};
```

```
    int size = sizeof(arr) / sizeof(arr[0]);

    printf("Array before sorting:\n");
    printArray(arr, size);

    quickSort(arr, 0, size - 1);

    printf("Array after sorting:\n");
    printArray(arr, size);

    return 0;
}
```

Output :



Experiment – 5

5 Write a program to implement Towers of Hanoi problem.

Code :

```
#include <stdio.h>

void towersOfHanoi(int numDisks, char source, char auxiliary, char destination) {
    if (numDisks == 1) {
        printf("Move disk 1 from %c to %c\n", source, destination);
        return;
    }

    towersOfHanoi(numDisks - 1, source, destination, auxiliary);
    printf("Move disk %d from %c to %c\n", numDisks, source, destination);
    towersOfHanoi(numDisks - 1, auxiliary, source, destination);
}

int main() {
    int numDisks = 3;
    char source = 'A';
    char auxiliary = 'B';
    char destination = 'C';

    printf("Towers of Hanoi solution:\n");
    towersOfHanoi(numDisks, source, auxiliary, destination);

    return 0;
}
```

Output:

```
Towers of Hanoi solution:
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```

Experiment – 6

6 Write a program to create a singly linked list and print the contents of it.

Code :

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


struct linkedlist
{
    int number;
    struct linkedlist *next;
};
typedef struct linkedlist node;
void main()
{
    int flag;
    node *head;
    void create(node*);
    void printlist(node *);
    head=(node *)malloc(sizeof(node));
    printf("\n\n\n Enter the element of the linkedlist");
    create(head);
      printf("\n\nThe elements in the linked list are:\n\n\n");
   printlist(head);
   printf("\n");



}
void create(node *list)
{
    scanf("%d",&list->number);
    if(list->number==-99)
    list->next=NULL;
    else
    {
        list->next=(node*)malloc(sizeof(node));
        create(list->next);
    }
}
void printlist(node *head)
{
    while(head!=NULL)
    {
        printf("%d,",head->number);
        head=head->next;
    }

    }
```

Output :

```
 Enter the element of the linkedlist1
2
35
6
7
-99


The elements in the linked list are:


1,2,35,6,7,-99,
```

Experiment – 7

7 Write a program to delete the first and last nodes of a singly linked list.

Code :

```
#include<stdio.h>
#include<stdlib.h>
struct linkedlist
 {
   int number;
   struct linkedlist *next;
 };
 typedef struct linkedlist node;
 void main()
 {

  int flag;
  node *head;
  void create(node*);
  void printlist(node*);
  node* delete_first(node*);
  void delete_last(node*);
  head = (node*)malloc(sizeof(node));
  printf("\n\n\n Enter the elements of the linked list , -99 to stop the creation of linked list\n\n");
  create(head);
  printf("\n");
  printf("\n\nThe elements in the linked list are:\n\n\n");
  printlist(head);
  head= delete_first(head);
  printf("\n\nThe elements in the linked list after deleting first node are:\n\n\n");
  printlist(head);
  delete_last(head);
  printf("\n\nThe elements in the linked list after deleting last node are :\n\n\n");
  printlist(head);
 }
 void create(node *list)
  {
```

```
    scanf("%d", &list->number);
    if(list->number == -99)
    list->next = NULL;
    else
     {
           list->next=(node*)malloc(sizeof(node));
           create(list->next);
     }
    }
    void printlist(node *list)
     {
          while(list!= NULL)
           {
             printf("%d  ",list->number);
             list=list->next;
           }
     }
node* delete_first(node *list)
              {
               list=list->next;
               return(list);
               }
             void delete_last(node *list)
               {
                node *temp;
                while(list->next!=NULL)
                {
                temp=list;
                list=list->next;
                }
               temp->next=NULL;
             }
```

Code:

```
 Enter the elements of the linked list , -99 to stop the creation of linked list

1
2
3
4
5
-99



The elements in the linked list are:


1  2  3  4  5  -99

The elements in the linked list after deleting first node are:


2  3  4  5  -99

The elements in the linked list after deleting last node are :


2  3  4  5
```

Experiment – 8

8 Write a program to add a new node in the beginning/ as the last node in a singly linked list

Code :

```
#include<stdio.h>
#include<stdlib.h>
struct linkedlist
  {
    int number;
    struct linkedlist *next;
  };
  typedef struct linkedlist node;
void main()
  {
   int flag;
   node *head;
   void create(node*);
   void printlist(node*);
   node* insert_first(node*);
   void insert_last(node*);
   head = (node*)malloc(sizeof(node));
   printf("\n\n\n Enter the elements of the linked list , -99 to stop the creation of linked list\n\n");
   create(head);
   printf("\n");
   printf("\n\nThe elements in the linked list are:\n\n\n");
   printlist(head);
   head= insert_first(head);
   printf("\n\nThe elements in the linked list after inserting a new node as first node are:\n\n\n");
   printlist(head);
   insert_last(head);
   printf("\n\nThe elements in the linked list after inserting a new node as last node are :\n\n\n");
   printlist(head);
  }

   void create(node *list)
    {
     scanf("%d", &list->number);
     if(list->number == -99)
     list->next = NULL;
     else
       {
           list->next=(node*)malloc(sizeof(node));
           create(list->next);
       }
     }

     void printlist(node *list)
       {
          while(list!= NULL)
           {
             printf("%d  ",list->number);
             list=list->next;
```

```
    }
}

    node* insert_first(node *list)
     {
     node *ne;
     printf("\n\nEnter the new element to be added as first node\n\n");
     ne=(node*)malloc(sizeof(node));
     ne->next=list;
     list=ne;
     scanf("%d", &ne->number);
     return(ne);
     }

 void insert_last(node *list)
     {
      node *ne;
      while(list!=NULL)
      {
      ne=list;
      list=list->next;
      }
     ne->next= (node*)malloc(sizeof(node));
     ne=ne->next;
     ne->next=NULL;
     printf("\n\nEnter the new element to be added as last node\n\n");
     scanf("%d",&ne->number);
    }
```

Output:

```
 Enter the elements of the linked list , -99 to stop the creation of linked list

1
2
3
4
-99


The elements in the linked list are:


1  2  3  4  -99

Enter the new element to be added as first node

8


The elements in the linked list after inserting a new node as first node are:


8  1  2  3  4  -99

Enter the new element to be added as last node

9


The elements in the linked list after inserting a new node as last node are :


8  1  2  3  4  -99  9
```

Experiment - 9

9 Write a program to reverse a singly linked list.

Code :

```
#include <stdio.h>
#include <stdlib.h>
struct linkedlist
{
   int number;
   struct linkedlist *next;
};
typedef struct linkedlist node;
void main()
{
   int flag;
   node *head;
   void create(node *);
   void printlist(node *);
   node *reverse(node *);
   head = (node *)malloc(sizeof(node));
   printf("\n\n\n Enter the elements of the linked list , -99 to stop the creation of linked list\n\n");
   create(head);
   printf("\n");
   printf("\n\nThe elements in the linked list are:\n\n\n");
   printlist(head);
   head = reverse(head);
   printf("\n\nThe elements in the linked list after its reversal are:\n\n\n");
   printlist(head);
}

void create(node *list)
{
   scanf("%d", &list->number);
   if (list->number == -99)
      list->next = NULL;
   else
   {
      list->next = (node *)malloc(sizeof(node));
      create(list->next);
   }
}

void printlist(node *list)
{
   while (list != NULL)
   {
      printf("%d  ", list->number);
```

```
        list = list->next;
    }
}

node *reverse(node *list)
{
    node *rev, *temp;
    rev = list;
    list = list->next;
    rev->next = NULL;
    while (list != NULL)
    {
        temp = list;
        list = list->next;
        temp->next = rev;
        rev = temp;
    }
    return (rev);
}
```

Output :

```
 Enter the elements of the linked list , -99 to stop the creation of linked list

1
2
3
4
5
-99



The elements in the linked list are:


1  2  3  4  5  -99

The elements in the linked list after its reversal are:


-99  5  4  3  2  1

...Program finished with exit code 0
Press ENTER to exit console.
```

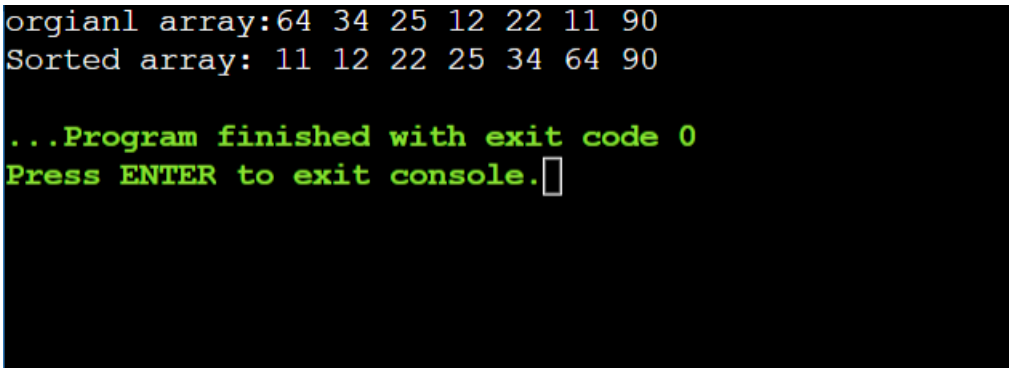
Experiment – 10

10 Write a program to implement Bubble Sort

Code :

```
#include <stdio.h>

void bubbleSort(int arr[], int n) {
    int i, j;

    for (i = 0; i < n - 1; i++) {
        // Last i elements are already in place
        for (j = 0; j < n - i - 1; j++) {
            // Swap if the element found is greater than the next element
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("orgianl array:");
     for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    bubbleSort(arr, n);
    printf("Sorted array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Output:


orgianl array:64 34 25 12 22 11 90
Sorted array: 11 12 22 25 34 64 90
...Program finished with exit code 0
Press ENTER to exit console.

Experiment-11

11 Write a program to implement insertion sort.

Code:

```
#include <stdio.h>

void insertionSort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        // Move elements of arr[0..i-1], that are greater than key, to one position ahead of their current
position
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }

        arr[j + 1] = key;
    }
}

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

int main() {
    int arr[] = {5, 2, 8, 12, 1, 6, 3, 9};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Before sorting:\n");
    printArray(arr, n);

    insertionSort(arr, n);

    printf("After sorting:\n");
    printArray(arr, n);

    return 0;
}
```

Output:

```
Before sorting:
5 2 8 12 1 6 3 9
After sorting:
1 2 3 5 6 8 9 12
```

Experiment-12

12 Write a program to implement Selection Sort.

Code:

```
#include <stdio.h>

void selectionSort(int arr[], int n) {
    int i, j, min_idx, temp;

    for (i = 0; i < n-1; i++) {
        min_idx = i;
        for (j = i+1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
                min_idx = j;
            }
        }
        // Swap the found minimum element with the first element
        temp = arr[min_idx];
        arr[min_idx] = arr[i];
        arr[i] = temp;
    }
}

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

int main() {
    int arr[] = {64, 25, 12, 22, 11};
    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: \n");
    printArray(arr, n);

    selectionSort(arr, n);

    printf("Sorted array: \n");
    printArray(arr, n);

    return 0;
}
```

Output:

```
n\gdb.exe' '--interpreter=mi'
Original array:
64 25 12 22 11
Sorted array:
11 12 22 25 64
PS C:\Users\ksaks>
```

Experiment-13

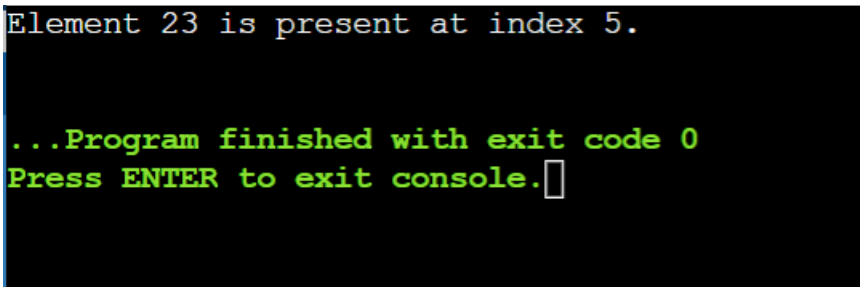
13 Write a program to implement Binary search.

Code:

```
#include <stdio.h>

int binarySearch(int arr[], int left, int right, int target) {
    while (left <= right) {
        int mid = left + (right - left) / 2;

        // Check if the target is present at the middle position
        if (arr[mid] == target) {
            return mid;
        }

        // If the target is greater, ignore the left half
        if (arr[mid] < target) {
            left = mid + 1;
        }

        // If the target is smaller, ignore the right half
        else {
            right = mid - 1;
        }
    }

    // If the target is not found
    return -1;
}

int main() {
    int arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91};
    int n = sizeof(arr) / sizeof(arr[0]);
    int target = 23;

    int result = binarySearch(arr, 0, n - 1, target);
```

```
    if (result == -1) {
        printf("Element %d is not present in the array.\n", target);
    }
    else {
        printf("Element %d is present at index %d.\n", target, result);
    }

    return 0;
}
```

Output:



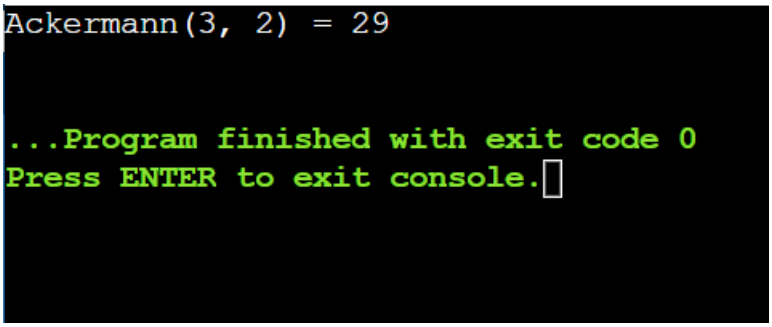
Experiment-14

14 Write a program to implement Ackermann's function.

Code:

```
#include <stdio.h>

int ackermann(int m, int n) {
    if (m == 0) {
        return n + 1;
    }
    else if (m > 0 && n == 0) {
        return ackermann(m - 1, 1);
    }
    else {
        return ackermann(m - 1, ackermann(m, n - 1));
    }
}

int main() {
    int m = 3;
    int n = 2;

    int result = ackermann(m, n);

    printf("Ackermann(%d, %d) = %d\n", m, n, result);

    return 0;
}
```

Output:


Ackermann(3, 2) = 29
...Program finished with exit code 0
Press ENTER to exit console.

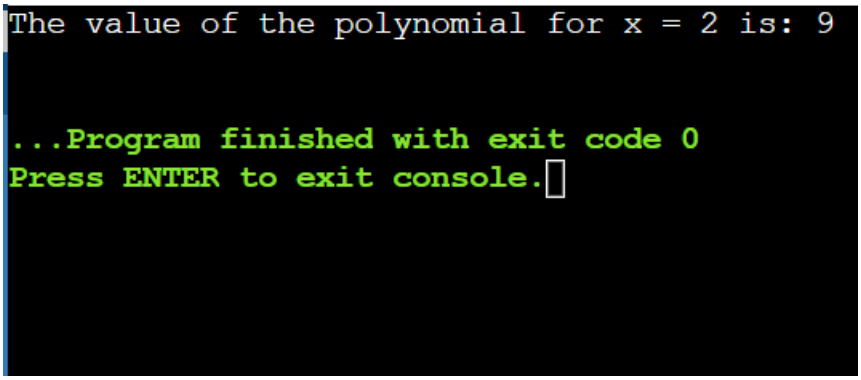
Expriment-15

15 Write a program to evaluate a Polynomial using Horner's Rule.

Code:

```
#include <stdio.h>

// Function to evaluate the polynomial using Horner's Rule
int evaluatePolynomial(int coefficients[], int n, int x) {
    int result = coefficients[0]; // Initialize the result with the coefficient of highest degree term

    for (int i = 1; i <= n; i++) {
        result = result * x + coefficients[i];
    }

    return result;
}

int main() {
    int coefficients[] = {3, -2, 1}; // Coefficients of the polynomial: 3x^2 - 2x + 1
    int degree = sizeof(coefficients) / sizeof(coefficients[0]) - 1; // Degree of the polynomial
    int x = 2; // Value of x for evaluation

    int result = evaluatePolynomial(coefficients, degree, x);

    printf("The value of the polynomial for x = %d is: %d\n", x, result);

    return 0;
}
```

Output:


The value of the polynomial for x = 2 is: 9
...Program finished with exit code 0
Press ENTER to exit console.