DEWAN V.S INSTITUTE OF ENGINEERING & TECHNOLOGY MEERUT

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SESSION-2025-26

LAB MANUAL

COURSE : B.Tech(CSE)
SEMESTER : IIIRD
SUBJECT : DS LAB
SUBJECTCODE : BCS-351

LISTOFEXPERIMENTS

COURSE:B.Tech **SEMESTER:** III **SESSION:**2025-26(ODD SEM)
BRANCH:CSE
SUBJECTCODE&NAME:BCS-351, (DATA STRUCTURE LAB)

SI.No.	NAME OF EXPERIMENT
1	Write a C program for addition of two matrix.
2	Write a C program for addition of two matrix.
3	Write a C program for insert the element in Array.
4	Write a C program for delete the element in Array.
5	Write a C program for search the element using Linear Search in Array.
6	Write a C program for search the element using Binary Search in Array.
7	Write a C program for Sort the element using Bubble Sort in Array.
8	Write a C program for Sort the element using Selection Sort in Array.
9	Write a C program for Sort the element using Insertion Sort in Array.
10	Write a C program for Sort the element using Quick Sort in Array.
11	Write a C program for Sort the element using Merge Sort in Array.
12	Write a C program for Sort the element using Heap Sort in Array.
13	Implementation of Stack using Array.
14	Implementation of Stack using Linked list.
15	Implementation of Queue using Array.
16	Implementation of Queue using Linked list.

Faculty Name- Ashwani Chauhan

Experiment-1

ProgramName:Write a C program for addition of two matrix.

```
#include <stdio.h>
int main() {
  int r, c, a[100][100], b[100][100], sum[100][100], i, j;
printf("Enter the number of rows (between 1 and 100): ");
scanf("%d", &r);
printf("Enter the number of columns (between 1 and 100): ");
scanf("%d", &c);

printf("\nEnter elements of 1st matrix:\n");
  for (i = 0; i< r; ++i)
    for (j = 0; j < c; ++j) {
printf("Enter element a%d%d: ", i + 1, j + 1);
scanf("%d", &a[i][j]);
    }

printf("Enter elements of 2nd matrix:\n");
  for (i = 0; i< r; ++i)
    for (j = 0; j < c; ++j) {
printf("Enter element b%d%d: ", i + 1, j + 1);
scanf("%d", &b[i][j]);
    }

  // adding two matrices
  for (i = 0; i< r; ++i)
    for (j = 0; j < c; ++j) {
      sum[i][j] = a[i][j] + b[i][j];
    }

  // printing the result
printf("\nSum of two matrices: \n");
  for (i = 0; i< r; ++i)
    for (j = 0; j < c; ++j) {
printf("%d   ", sum[i][j]);
      if (j == c - 1) {
printf("\n\n");
       }
    }

  return 0;
```

```
}
```

OUTPUT:

Enter the number of rows (between 1 and 100): 1
Enter the number of columns (between 1 and 100): 1

Enter elements of 1st matrix:
Enter element a11: 3
Enter elements of 2nd matrix:
Enter element b11: 4

Sum of two matrices:
7

Experiment-2

ProgramName:Write a C program for subtraction of two matrix.

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int mat1[3][3], mat2[3][3], matSub[3][3], i, j;
printf("Enter First 3*3 Matrix Elements: ");
for(i=0; i<3; i++)
   {
for(j=0; j<3; j++)
scanf("%d", &mat1[i][j]);
   }
printf("Enter Second 3*3 Matrix Elements: ");
for(i=0; i<3; i++)
   {
for(j=0; j<3; j++)
scanf("%d", &mat2[i][j]);
   }
for(i=0; i<3; i++)
   {
```

```
for(j=0; j<3; j++)
matSub[i][j] = mat1[i][j] - mat2[i][j];
   }
printf("\nThe Subtraction Result is:\n");
for(i=0; i<3; i++)
   {
for(j=0; j<3; j++)
printf("%d  ", matSub[i][j]);
printf("\n");
   }
getch();
   return 0;
}
```

OUTPUT:

Enter First 3*3 Matrix Elements: 2
1
3
2
4
4
5
6
3
Enter Second 3*3 Matrix Elements: 2
3
4
6
2
1
5
4
5

The Subtraction Result is:
0 -2 -1
-4 2 3
0 2 -2

Experiment-3

Program Name: Write a C program for Insert the element in Array.

```
#include <stdio.h>

int main ()
{
   int array[50], position, c, n, value;

printf("Enter number of elements in the array\n");
scanf("%d", &n);

printf("Enter %d elements\n", n);

   for (c = 0; c < n; c++)
scanf("%d", &array[c]);

printf("Please enter the location where you want to insert an new element\n");
scanf("%d", &position);

printf("Please enter the value\n");
scanf("%d", &value);

   for (c = n - 1; c >= position - 1; c--)
      array[c+1] = array[c];

   array[position-1] = value;

printf("Resultant array is\n");

   for (c = 0; c <= n; c++)
printf("%d\n", array[c]);

   return 0;
}
```

OUTPUT:

Enter number of elements in the array
5
Enter 5 elements
2
4
5

6
7
Please enter the location where you want to insert an new element
3
Please enter the value
8
Resultant array is
2
4
8
5
6
7

Experiment-4

ProgramName:Write a C program for Delete the element in Array.

```
#include <stdio.h>

int main()
{
   int array[100], position, c, n;

printf("Enter number of elements in array\n");
scanf("%d", &n);

printf("Enter %d elements\n", n);

   for ( c = 0 ; c < n ; c++ )
scanf("%d", &array[c]);

printf("Enter the location where you wish to delete element\n");
scanf("%d", &position);

   if ( position>= n+1 )
printf("Deletion not possible.\n");

   else
   {
      for ( c = position - 1 ; c < n - 1 ; c++ )
      array[c] = array[c+1];

printf("Resultant array is\n");
```

```
    for( c = 0 ; c < n - 1 ; c++ )
    printf("%d\n", array[c]);
       }
       return 0;
    }
```

OUTPUT:

Enter number of elements in array
5
Enter 5 elements
3
6
7
8
9
Enter the location where you wish to delete element
2
Resultant array is
3
7
8
9

Experiment-5

ProgramName:Write a C program for search the element using Linear Search in Array.

```
#include <stdio.h>
int main()
{
  int array[100], search, c, n;
  printf("Enter number of elements in array\n");
  scanf("%d", &n);
  printf("Enter %d integer(s)\n", n);
  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);
  printf("Enter a number to search\n");
  scanf("%d", &search);
  for (c = 0; c < n; c++)
  {
    if (array[c] == search)    /* If required element is found */
```

```
    {
      printf("%d is present at location %d.\n", search, c+1);
      break;
    }
  }
  if (c == n)
    printf("%d isn't present in the array.\n", search);
  return 0;
}
```

OUTPUT:

Enter number of elements in array
6
Enter 6 integer(s)
2
3
5
6
7
8
Enter a number to search
5
5 is present at location 3.

Experiment-6

ProgramName:Write a C program for search the element using Binary Search in Array.

```
#include <stdio.h>
int main()
{
  int c, first, last, middle, n, search, array[100];
  printf("Enter number of elements\n");
  scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);
  printf("Enter value to find\n");
  scanf("%d", &search);
  first = 0;
  last = n - 1;
  middle = (first+last)/2;
  while (first <= last) {
```

```
middle = (first + last)/2;
    if (array[middle] < search)
      first = middle + 1;
    else if (array[middle] == search) {
      printf("%d found at location %d.\n", search, middle+1);
      break;
    }
    else
      last = middle – 1;
  }
  if (first > last)
    printf("Not found! %d isn't present in the list.\n", search);
  return 0;
}
```

OUTPUT:
Enter number of elements
4
Enter 4 integers
2
3
4
5
Enter value to find
4
4 found at location 3.

Experiment- 7

ProgramName:Write a C program for Sort the element using Bubble Sort in Array.

```
#include <stdio.h>

int main()
{
  int array[100], n, c, d, swap;

printf("Enter number of elements\n");
scanf("%d", &n);

printf("Enter %d integers\n", n);
```

```
  for (c = 0; c < n; c++)
scanf("%d", &array[c]);

  for (c = 0 ; c < n - 1; c++)
  {
    for (d = 0 ; d < n - c - 1; d++)
    {
      if (array[d] > array[d+1]) /* For decreasing order use '<' instead of '>' */
      {
        swap       = array[d];
        array[d]   = array[d+1];
        array[d+1] = swap;
      }
    }
  }

printf("Sorted list in ascending order:\n");

  for (c = 0; c < n; c++)
printf("%d\n", array[c]);

  return 0;
}
```

OUTPUT:

Enter number of elements
6
Enter 6 integers
3
65
8
9
3
9
Sorted list in ascending order:
3
3
8
9
9
65

Experiment- 8

Program Name:Write a C program for Sort the element using Selection Sort in Array.

```
#include<stdio.h>

int main()
{
  int array[100], n, c, d, position, t;

printf("Enter number of elements\n");
scanf("%d", &n);

printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
scanf("%d", &array[c]);

  for (c = 0; c < (n - 1); c++) // finding minimum element (n-1) times
  {
    position = c;

    for (d = c + 1; d < n; d++)
    {
      if (array[position] > array[d])
        position = d;
    }
    if (position != c)
    {
      t = array[c];
      array[c] = array[position];
      array[position] = t;
    }
  }

printf("Sorted list in ascending order:\n");

  for (c = 0; c < n; c++)
printf("%d\n", array[c]);

  return 0;
}
```

OUTPUT:

Enter number of elements
7
Enter 7 integers
4
2
7
8
4
3
9
Sorted list in ascending order:
2
3
4
4
7
8
9

Experiment- 9

Program Name:Write a C program for Sort the element using Insertion Sort in Array.

```
#include <stdio.h>

int main()
{
  int n, array[1000], c, d, t, flag = 0;

printf("Enter number of elements\n");
scanf("%d", &n);

printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
scanf("%d", &array[c]);

  for (c = 1 ; c <= n - 1; c++) {
    t = array[c];

    for (d = c - 1 ; d >= 0; d--) {
      if (array[d] > t) {
        array[d+1] = array[d];
        flag = 1;
      }
      else
break;
    }
    if (flag)
      array[d+1] = t;
  }

printf("Sorted list in ascending order:\n");

  for (c = 0; c <= n - 1; c++) {
printf("%d\n", array[c]);
  }

  return 0;
}
```

OUTPUT:

Enter number of elements
7
Enter 7 integers
2
6
3
1
4
8
5
Sorted list in ascending order:
1
2
3
4
5
6
8

Experiment- 10

Program Name:Write a C program for Sort the element using Quick Sort in Array.

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

int quickSort(int *arr, int low, int high)
{
   int i = low, j = high;
   int pivot = arr[(low + high) / 2];
   while (i<= j)
   {
      while (arr[i] < pivot)
i++;
      while (arr[j] > pivot)
         j--;
      if (i<= j)
      {
         int temp = arr[i];
arr[i] = arr[j];
arr[j] = temp;
i++;
```

```
            j--;
        }
    }
    if (low < j)
quickSort(arr, low, j);
    if (i< high)
quickSort(arr, i, high);
    return 0;
}

int main(void)
{
puts("Enter the number of elements in the array: ");
    int n;
scanf("%d", &n);
    int arr[n];
puts("Enter the elements of the array: ");
    for (int i = 0; i< n; i++)
    {
printf("arr[%d]: ", i);
scanf("%d", &arr[i]);
    }
    int low = 0;
    int high = n - 1;
    int pivot = arr[high];
    int k = low - 1;
    for (int j = low; j < high; j++)
    {
        if (arr[j] <= pivot)
        {
            k++;
            int temp = arr[k];
arr[k] = arr[j];
arr[j] = temp;
        }
    }
    int temp = arr[k + 1];
arr[k + 1] = arr[high];
arr[high] = temp;
```

```
    int pi = k + 1;
quickSort(arr, low, pi - 1);
quickSort(arr, pi + 1, high);
puts("The sorted array is: ");
    for (int i = 0; i< n; i++)
    {
printf("%d ", arr[i]);
    }
    return 0;
}
```

OUTPUT:

Enter the number of elements in the array:
7
Enter the elements of the array:
arr[0]: 3
arr[1]: 4
arr[2]: 1
arr[3]: 9
arr[4]: 7
arr[5]: 2
arr[6]: 8
The sorted array is:
1 2 3 4 7 8 9

Experiment- 11

Program Name:Write a C program for Sort the element using Merge Sort in Array.

```
#include <stdio.h>

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

```
}

void merge(int A[], int mid, int low, int high)
{
   int i, j, k, B[100];
i = low;
   j = mid + 1;
   k = low;

   while (i<= mid && j <= high)
   {
      if (A[i] < A[j])
      {
         B[k] = A[i];
i++;
         k++;
      }
      else
      {
         B[k] = A[j];
j++;
         k++;
      }
   }
   while (i<= mid)
   {
      B[k] = A[i];
      k++;
i++;
   }
   while (j <= high)
   {
      B[k] = A[j];
      k++;
j++;
   }
   for (int i = low; i<= high; i++)
   {
      A[i] = B[i];
```

```
    }

}

void mergeSort(int A[], int low, int high){
    int mid;
    if(low<high){
        mid = (low + high) /2;
mergeSort(A, low, mid);
mergeSort(A, mid+1, high);
merge(A, mid, low, high);
    }
}

int main()
{
    // int A[] = {9, 14, 4, 8, 7, 5, 6};
    int A[] = {9, 1, 4, 14, 4, 15, 6};
    int n = 7;
printArray(A, n);
mergeSort(A, 0, 6);
printArray(A, n);
    return 0;
}
```

OUTPUT:

9 1 4 14 4 15 6
1 4 4 6 9 14 15

Experiment- 12

Program Name:Write a C program for Sort the element using Heap Sort in Array.

```
#include<stdio.h>

// function prototyping
void heapify(int*,int, int);
void heapsort(int*, int);
void print_array(int*, int);

int main()
{
   int arr[] = { 10, 30, 5, 63, 22, 12, 56, 33 };
   int n = sizeof(arr) / sizeof(arr[0]);

printf("\nArray before sorting:\n");
print_array(arr, n);

heapsort(arr, n);

printf("\n\nArray after sorting:\n");
print_array(arr, n);

   return 0;
}

/* sorts the given array of n size */
void heapsort(int* arr, int n)
{
   // build the binary max heap
   for (int i = n / 2 - 1; i>= 0; i--)
   {
heapify(arr, n, i);
   }

   // sort the max heap
   for (int i = n - 1; i>= 0; i--)
   {
      // swap the root node and the last leaf node
      int temp = arr[i];
arr[i] = arr[0];
```

```
arr[0] = temp;

        // againheapify the max heap from the root
heapify(arr, i, 0);
    }
}

/* heapify the subtree with root i */
void heapify(int* arr, int n, int i)
{
    // store largest as the root element
    int largest = i;

    int left = 2 * i + 1;
    int right  = 2 * i + 2;

    // now check whether the right and left right is larger than the root or not
    if (left < n &&arr[left] >arr[largest])
    {
        largest = left;
    }

    if (right < n &&arr[right] >arr[largest])
    {
        largest = right;
    }

    // if the root is smaller than the children then swap it with the largest children's value
    if (largest != i)
    {
        int temp = arr[i];
arr[i] = arr[largest];
arr[largest] = temp;

        // again,heapify that side of the heap where the root has gone
heapify(arr, n, largest);
    }
}

/* printf the array */
void print_array(int* arr, int n)
{
    for (int i = 0; i< n; i++)
```

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

OUTPUT:

Array before sorting:
10 30 5 63 22 12 56 33

Array after sorting:
5 10 12 22 30 33 56 63

Experiment- 13

Program Name: **Implementation of Stack using Array.**

```
#include <stdio.h>

// Function declarations
void push();
void pop();
void peek();

// Taking stack array of size 50
int stack[50],i,j,option=0,n,top=-1;
void main ()
{
//  Size of stack
printf("Enter the number of elements in the stack ");
scanf("%d",&n);
printf("Stack implementation using array\n");

//     Taking user input for the operation to be performed
while(option != 4)
   {
printf("Chose one from the below options...\n");
printf("\n1.Push\n2.Pop\n3.Peek\n4.Exit");
printf("\n Enter your option \n");
scanf("%d",&option);
       switch(option)
       {
          case 1:
          {
push();
break;
          }
          case 2:
          {
pop();
break;
          }
          case 3:
```

```
            {
peek();
break;
            }
            case 4:
            {
printf("Exiting");
break;
            }
            default:
            {
printf("Please Enter valid option");
            }
        };
    }
}

// Push operation function
void push ()
{
    int value;
    if (top == n )
printf("\n Overflow");
    else
    {
printf("Enter the value?");
scanf("%d",&value);
        top = top +1;
        stack[top] = value;
    }
}

// Pop operation function
void pop ()
{
if(top == -1)
printf("Underflow");
    else
    top = top -1;
}

// peek operation function
void peek()
{
    for (i=top;i>=0;i--)
```

```
    {
printf("%d\n",stack[i]);
    }
if(top == -1)
    {
printf("Stack is empty");
    }
}
```

OUTPUT:

Enter the number of elements in the stack 3
Stack implementation using array
Chose one from the below options...

1.Push
2.Pop
3.Peek
4.Exit
Enter your option
1
Enter the value?2
Chose one from the below options...

1.Push
2.Pop
3.Peek
4.Exit
Enter your option
1
Enter the value?5
Chose one from the below options...

1.Push
2.Pop
3.Peek
4.Exit
Enter your option
1
Enter the value?4
Chose one from the below options...

1.Push
2.Pop
3.Peek
4.Exit
Enter your option
3
4
5
2
Chose one from the below options...

1.Push

2.Pop
3.Peek
4.Exit
Enter your option

Experiment- 14

Program Name:Implementation of Stack using Linked List.

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

// Structure to create a node with data and the next pointer
struct node {
   int info;
   struct node *ptr;
}*top,*top1,*temp;

int count = 0;
// Push() operation on a  stack
void push(int data) {
    if (top == NULL)
   {
      top =(struct node *)malloc(1*sizeof(struct node));
      top->ptr = NULL;
      top->info = data;
   }
   else
   {
      temp =(struct node *)malloc(1*sizeof(struct node));
      temp->ptr = top;
      temp->info = data;
      top = temp;
   }
   count++;
printf("Node is Inserted\n\n");
}

int pop() {
    top1 = top;
```

```
    if (top1 == NULL)
    {
printf("\nStack Underflow\n");
        return -1;
    }
    else
        top1 = top1->ptr;
    int popped = top->info;
    free(top);
    top = top1;
    count--;
    return popped;
}

void display() {
    // Display the elements of the stack
    top1 = top;

    if (top1 == NULL)
    {
printf("\nStack Underflow\n");
return;
    }

printf("The stack is \n");
    while (top1 != NULL)
    {
printf("%d--->", top1->info);
        top1 = top1->ptr;
    }
printf("NULL\n\n");

}

int main() {
    int choice, value;
printf("\nImplementation of Stack using Linked List\n");
    while (1) {
```

```
printf("\n1. Push\n2. Pop\n3. Display\n4. Exit\n");
printf("\nEnter your choice : ");
scanf("%d", &choice);
        switch (choice) {
        case 1:
printf("\nEnter the value to insert: ");
scanf("%d", &value);
            push(value);
break;
        case 2:
printf("Popped element is :%d\n", pop());
break;
        case 3:
display();
break;
        case 4:
exit(0);
break;
        default:
printf("\nWrong Choice\n");
        }
    }
}
```

OUTPUT:

Implementation of Stack using Linked List

1. Push
2. Pop
3. Display
4. Exit

Enter your choice : 1

Enter the value to insert: 3
Node is Inserted

1. Push
2. Pop

3. Display
4. Exit

Enter your choice : 4

Experiment- 15

Program Name:Implementation of Queue using Array.

```
#include <stdio.h>

#define MAX 50

void insert();
void delete();
void display();
int queue_array[MAX];
int rear = - 1;
int front = - 1;
main()
{
   int choice;
   while (1)
   {
printf("1.Insert element to queue \n");
printf("2.Delete element from queue \n");
printf("3.Display all elements of queue \n");
printf("4.Quit \n");
printf("Enter your choice : ");
scanf("%d", &choice);
       switch (choice)
       {
          case 1:
insert();
break;
          case 2:
delete();
break;
          case 3:
display();
break;
          case 4:
exit(1);
```

```
            default:
printf("Wrong choice \n");
        } /* End of switch */
    } /* End of while */
} /* End of main() */

void insert()
{
    int add_item;
    if (rear == MAX - 1)
printf("Queue Overflow \n");
    else
    {
        if (front == - 1)
        /*If queue is initially empty */
        front = 0;
printf("Inset the element in queue : ");
scanf("%d", &add_item);
        rear = rear + 1;
queue_array[rear] = add_item;
    }
} /* End of insert() */

void delete()
{
    if (front == - 1 || front > rear)
    {
printf("Queue Underflow \n");
return ;
    }
    else
    {
printf("Element deleted from queue is : %d\n", queue_array[front]);
        front = front + 1;
    }
} /* End of delete() */

void display()
{
    int i;
    if (front == - 1)
printf("Queue is empty \n");
    else
    {
```

```
printf("Queue is : \n");
       for (i = front; i<= rear; i++)
printf("%d ", queue_array[i]);
printf("\n");
   }
} /* End of display() */
```

OUTPUT:

1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 3
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :
3
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 4

Experiment- 16

Program Name:Implementation of Queue using Linked List.

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

// Structure to create a node with data and the next pointer
struct node {
   int data;
   struct node * next;
};

struct node * front = NULL;
struct node * rear = NULL;
```

```
// Enqueue() operation on a queue
void enqueue(int value) {
   struct node * ptr;
ptr = (struct node * ) malloc(sizeof(struct node));
ptr - > data = value;
ptr - > next = NULL;
   if ((front == NULL) && (rear == NULL)) {
      front = rear = ptr;
   } else {
      rear - > next = ptr;
      rear = ptr;
   }
printf("Node is Inserted\n\n");
}

// Dequeue() operation on a queue
int dequeue() {
   if (front == NULL) {
printf("\nUnderflow\n");
      return -1;
   } else {
      struct node * temp = front;
      int temp_data = front - >data;
      front = front - >next;
      free(temp);
      return temp_data;
   }
}

// Display all elements of the queue
void display() {
   struct node * temp;
   if ((front == NULL) && (rear == NULL)) {
printf("\nQueue is Empty\n");
   } else {
printf("The queue is \n");
      temp = front;
      while (temp) {
```

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

int main() {
    int choice, value;
printf("\nImplementation of Queue using Linked List\n");
    while (choice != 4) {
printf("1.Enqueue\n2.Dequeue\n3.Display\n4.Exit\n");
printf("\nEnter your choice : ");
scanf("%d", & choice);
        switch (choice) {
            case 1:
printf("\nEnter the value to insert: ");
scanf("%d", & value);
                enqueue(value);
break;
            case 2:
printf("Popped element is :%d\n", dequeue());
break;
            case 3:
display();
break;
            case 4:
exit(0);
break;
            default:
printf("\nWrong Choice\n");
        }
    }
    return 0;
}
```

OUTPUT:

Implementation of Queue using Linked List
1.Enqueue

2.Dequeue
3.Display
4.Exit

Enter your choice : 1

Enter the value to insert: 2
Node is Inserted

1.Enqueue
2.Dequeue
3.Display
4.Exit

Enter your choice : 1

Enter the value to insert: 4
Node is Inserted

1.Enqueue
2.Dequeue
3.Display
4.Exit

Enter your choice : 1

Enter the value to insert: 7
Node is Inserted

1.Enqueue
2.Dequeue
3.Display
4.Exit

Enter your choice : 3
The queue is
2--->4--->7--->NULL

1.Enqueue
2.Dequeue
3.Display
4.Exit

Enter your choice : 4