**Ex. No.: 01 1.A Matrix Multiplication using 2-D Array Date:**

**Aim:** To Implement Program for 2-D Matrix Multiplication using Arrays

**Pseudo Code :**

**1]** Algorithm readMatrix (size, matrix [][])

1. assign i and j with 0
2. loop (i < size)
3. loop (j < size)
4. get dataIn for matrix[i][j] from user
5. increment j
6. end loop
7. increment i
8. end loop

end readMatrix

**2]** Algorithm multiplyMatrices (size, m1 [][], m2 [][], m3 [][])

1. assign i, j and k with 0
2. loop (i < size)
3. loop (j < size)
4. assign C [i] [j] with 0
5. loop (k < size)

1. assign C[i][j] with C[i][j] + A[i][k] \* B[k][j];

2. increment k

1. end loop
2. increment j
3. end loop
4. increment i
5. end loop

end multiplyMatrices

**3]** Algorithm printMatrix (size, matrix[][])

1. assign i and j with 0
2. loop (i < size)
3. loop (j < size)
4. print data of matrix[i][j]
5. increment j
6. end loop
7. increment i
8. end loop

end printMatrix

**Algorithm :**

1. Start the program.
2. Declare variables: n, A[MAX][MAX], B[MAX][MAX], C[MAX][MAX]
3. Display "Enter Size Of The Matrices (MAX: 10): "
4. Read input n using scanf() function
5. If n is less than or equal to 0, display "(Invalid) Enter a Non-Zero Number: " and go to step 4.
6. Display "Enter Elements For First Matrix: "
7. Call readMatrix(n, A) function to read n x n elements of A matrix.
8. Display "Enter Elements For Second Matrix: "
9. Call readMatrix(n, B) function to read n x n elements of B matrix.
10. Call multiplyMatrices(n, A, B, C) function to multiply A and B matrices and store the result in C matrix.
11. Display "Result After Multiplying Two Matrices: "
12. Call printMatrix(n, C) function to print n x n elements of C matrix.
13. End the program.

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include<stdio.h>

#define MAX 10

void readMatrix(int n,int matrix[][MAX])

{

int i,j;

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

printf("M[%d][%d]:",i,j);

scanf("%d",&matrix[i][j]);

}

}

}

void multiplyMatrices(int n,int A[][MAX],int B[][MAX],int C[][MAX])

{

int i,j,k;

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

C[i][j]=0;

for(k=0;k<n;k++)

{

C[i][j]+=A[i][k]\*B[k][j];

}

}

}

}

void printMatrix(int n,int matrix[][MAX])

{

int i,j;

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

printf("%d",matrix[i][j]);

}

printf("\n");

}

}

int main()

{

int n, A[MAX][MAX],B[MAX][MAX],C[MAX][MAX];

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No: 2028\n\n”);

printf("Enter size of the Matrices(MAX:10):");

do

{

scanf("%d",&n);

if(n<=0)

printf("(invalid)Enter a non-zero number:");

}

while(n<=0);

printf("\nEnter Elements for first Matrix:\n");

readMatrix(n,A);

printf("\nEnter Elements for second Matrix:\n");

readMatrix(n,B);

multiplyMatrices(n,A,B,C);

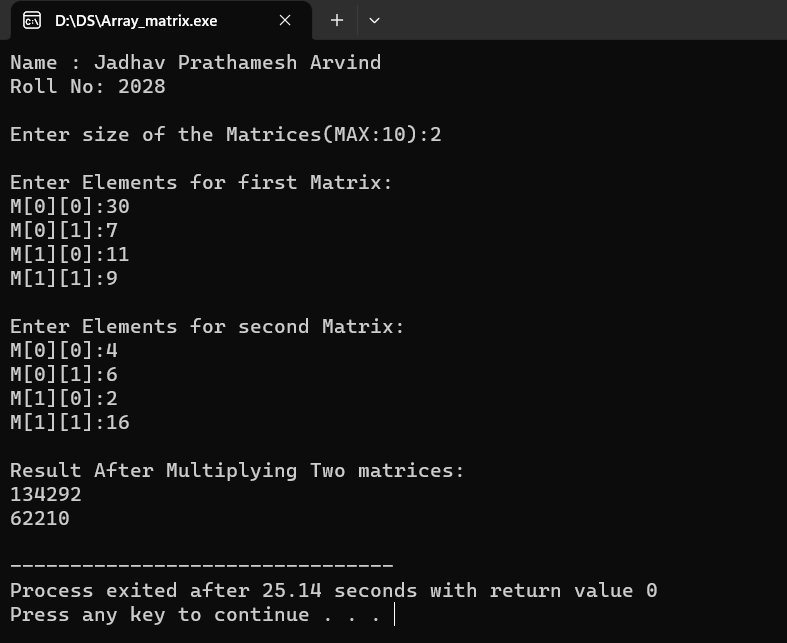
printf("\nResult After Multiplying Two matrices:\n");

printMatrix(n,C);

return 0;

}

* **Output :**

****

* **Result:**

Thus, the program for program for 2-D matrix multiplication using arrays was executed successfully.

**Ex. No.: 01 1.B Program for Factorial using Recursion Date:**

**Aim:** To Implement Program for factorial using recursion

**Algorithm:**

**1]** Algorithm factorial (n)

1. if (n equals 0)
2. return
3. return n \* factorial (n - 1)
4. end if

end factorial

1. Start the program.
2. Declare variable n.
3. Display "Enter a positive integer: "
4. Read input n using scanf() function.
5. If n is less than 0, display "(Invalid) Enter a positive integer: " and go to step 4.
6. Call factorial(n) function to calculate the factorial of n.
7. Display "The factorial of n is result", where result is the value returned by the factorial(n) function.
8. End the program

* **Program :**

/\* Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

int factorial(int n) {

if (n == 0) {

return 1;

}

return n \* factorial(n - 1);

}

int main() {

int n;

printf("Name : Jadhav Prathamesh Arvind\n");

printf("Roll No: 2028\n\n");

printf("Enter a positive integer: ");

do {

scanf("%d", & n);

if (n < 0) {

printf("(Invalid) Enter a positive integer: ");

}

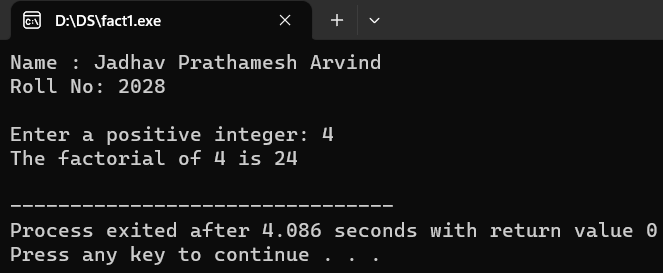
} while (n < 0);

printf("The factorial of %d is %d\n", n, factorial(n));

return 0;

}

* **Output:**



* **Result:**

Thus, the program for factorial using recursion was executed successfully.

**Ex. No.: 01 1.C Swapping of Two Numbers using Pointer Date:**

**Aim:** To Implement Program for Swapping of Two Numbers using Pointer

**Algorithm:**

**1]** Algorithm swap (pointer\_a, pointer\_b)

1. assign pointer\_a with pointer\_a + pointer\_b
2. assign pointer\_b with pointer\_a - pointer\_b
3. assign pointer\_a with pointer\_a - pointer\_b

end swap

1. Start the program.
2. Display "Enter Two Numbers: ".
3. Read input a and b using scanf() function.
4. Display "Before Swapping: a = %d, b = %d\n", a, b.
5. Call swap(&a, &b) function to swap the values of a and b using their memory addresses.
6. Display "After Swapping: a = %d, b = %d\n", a, b.
7. End the program

* **Program:**

/\* Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

void swap(int \*a, int \*b) {

\*a = \*a + \*b;

\*b = \*a - \*b;

\*a = \*a - \*b;

}

int main() {

int a, b;

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No : 2028\n\n”);

printf("Enter Two Numbers: ");

scanf("%d%d", &a, &b);

printf("Before Swapping: a = %d, b = %d\n", a, b);

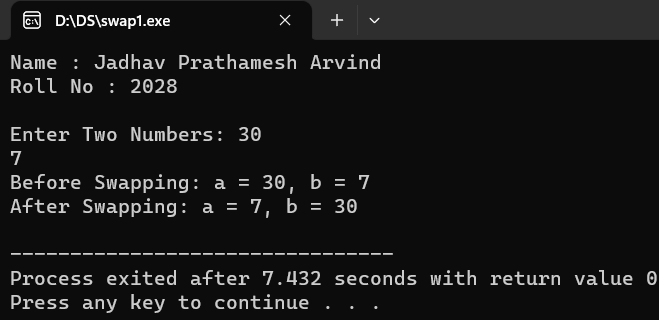
swap(&a, &b);

printf("After Swapping: a = %d, b = %d\n", a, b);

return 0;

}

* **Output :**

****

* **Result:**

Thus, the program for swapping of two numbers using pointers was executed successfully.

**Ex. No.: 01 1.D Student Database using Structures Date:**

**Aim:** To Implement Program for Student Database using Structures

**Algorithm:**

**1]** Algorithm readStudent (struct Student \*s)

1. get student Name and store in s->name
2. get student Roll\_No and store in s->roll
3. get student CPI and store in s->marks

end readStudent

**2]** Algorithm printStudent (struct Student s)

1. display s->name
2. display s->roll
3. display s->marks

end printStudent

1. Start the program.
2. Declare struct variable: students[MAX\_STUDENTS], n, i
3. Display "Enter The Number Of Students (MAX: MAX\_STUDENTS): "
4. Read input n using scanf() function
5. If n is less than or equal to 0, display "(Invalid) Enter a Non-Zero Number: " and go to step 4.
6. Use for loop to read information for each student by calling readStudent() function.
7. Use another for loop to print information for each student by calling printStudent() function.
8. End the program

* **Program:**

/\* Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#define MAX\_STUDENTS 10

struct Student {

char name[50];

int roll;

float marks;

};

void readStudent(struct Student \*s) {

printf("Enter Student First name: ");

scanf("%s", s->name);

printf("Enter Student roll: ");

scanf("%d", &s->roll);

printf("Enter Student CPI: ");

scanf("%f", &s->marks);

}

void printStudent(struct Student s) {

printf("Name: %s\n", s.name);

printf("Roll: %d\n", s.roll);

printf("CPI: %.2f\n", s.marks);

}

int main() {

struct Student students[MAX\_STUDENTS];

int n, i;

printf(“Name : Jadhav Prathamesh Arvind \n”);

printf(“Roll No : 2028\n\n”);

printf("Enter The Number Of Students (MAX: %d): ", MAX\_STUDENTS);

do {

scanf("%d", &n);

if (n <= 0) {

printf("(Invalid) Enter a Non-Zero Number: ");

}

} while (n <= 0);

for (i = 0; i < n; i++) {

printf("\nEnter Information For Student %d:\n", i+1);

readStudent(&students[i]);

}

for (i = 0; i < n; i++) {

printf("\nInformation Of Student %d:\n", i+1);

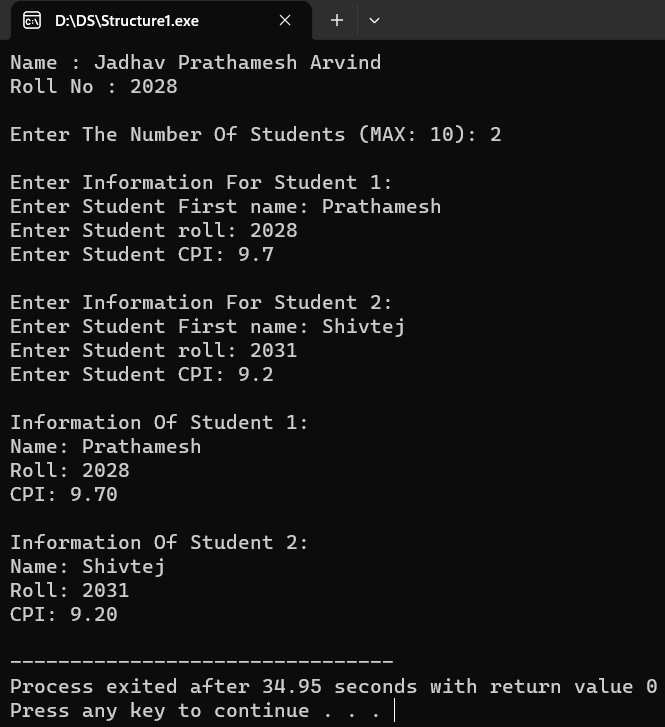
printStudent(students[i]);

}

return 0;

}

* **Output :**

****

* **Result:**

Thus, the program for student database using structure was executed successfully.

**Ex. No.: 02 Implementation of Singly Linked List Date:**

**Aim:** To Implement Program for Singly Linked List.

**Algorithm:**

**1]** Algorithm insertStart (dataIn)

1. allocate (newNode)
2. assign newNode data with dataIn
3. assign newNode next with headNode
4. assign headNode to newNode

end InsertStart

**2]** Algorithm insertEnd (dataIn)

1. allocate (newNode)
2. assign newNode data with dataIn
3. assign newNode next with null
4. if (headNode null)
5. assign headNode with newNode
6. return
7. end if
8. assign lastNode with headNode
9. loop (lastNode next not NULL)
10. assign lastNode with lastNode next
11. end loop
12. assign lastNode next with newNode

end insertEnd

**3]** Algorithm deleteStart ()

1. if (headNode null)
2. return
3. end if
4. assign temp with head
5. assign head with head next
6. free (temp)

end deleteStart

**4]** Algorithm deleteEnd ()

1. assign temp with head
2. assign prev with null
3. if (temp null)
4. return
5. end if
6. if (temp next null)
7. assign head with null
8. return
9. end if
10. loop (temp next not null)
11. assign prev with temp
12. assign temp with temp next
13. end loop
14. assign previous next with null

enddeleteEnd

**5]** Algorithm search (dataIn)

1. assign node with head
2. assign count with 1 and flag with 0
3. if (headNode null)
4. return
5. end if
6. loop (node not null)
7. if (node data equals dataIn)
8. assign flag with 1
9. display value of count
10. end if
11. assign node with node next
12. increment count
13. end loop
14. if (flag equals 0)
15. return 0
16. end if

end search

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include<stdio.h>

#include<stdlib.h>

struct Node {

int data;

struct Node \* next;

};

struct Node \* head = NULL;

void insertStart(int data) {

struct Node \* newNode = (struct Node \* ) malloc(sizeof(struct Node));

newNode -> data = data;

newNode -> next = head;

head = newNode;

}

void insertEnd(int data) {

struct Node \* newNode = (struct Node \* ) malloc(sizeof(struct Node));

newNode -> data = data;

newNode -> next = NULL;

if (head == NULL) {

head = newNode;

return;

}

struct Node \* last = head;

while (last -> next != NULL) {

last = last -> next;

}

last -> next = newNode;

}

void deleteStart() {

if (head == NULL) {

printf("List is Empty\n");

return;

}

struct Node \* temp = head;

head = head -> next;

free(temp);

printf("Node Deleted\n");

}

void deleteEnd() {

struct Node \* temp = head, \* previous;

if (head == NULL) {

printf("List is Empty\n");

return;

}

if (temp -> next == NULL) {

printf("Node Deleted\n");

head = NULL;

return;

}

while (temp -> next != NULL) {

previous = temp;

temp = temp -> next;

}

previous -> next = NULL;

printf("Node Deleted\n");

}

void search(int data) {

struct Node \* node = head;

int count = 1, flag = 0;

if (node == NULL) {

printf("List is Empty\n");

return;

}

flag = 0;

while (node != NULL) {

if (node -> data == data) {

printf("[[Node->Data: %d]] Found at %d Position in List", node -> data, count);

flag = 1;

}

node = node -> next;

}

if (flag == 0)

printf("Node not found");

printf("\n");

}

void display() {

struct Node \* ptr = head;

if (ptr == NULL) {

printf("List is Empty\n");

} else {

printf("List is: ");

while (ptr != NULL) {

printf(" %d ", ptr -> data);

ptr = ptr -> next;

}

printf("\n");

}

}

int getData() {

int data;

printf("Enter Element: ");

scanf("%d", & data);

return data;

}

int main() {

int ch;

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No :2028\n\n”);

printf("\*\*\*\*\*Singly Linked List\*\*\*\*\*\n\n");

printf("1. Insert Start\n2. Insert End\n3. Delete Start\n4. Delete End\n5. Search\n6. Display\n7. Exit");

do {

printf("\nEnter Your Choice: ");

scanf("%d", & ch);

if (ch == 1) {

insertStart(getData());

} else if (ch == 2) {

insertEnd(getData());

} else if (ch == 3) {

deleteStart();

} else if (ch == 4) {

deleteEnd();

} else if (ch == 5) {

search(getData());

} else if (ch == 6) {

display();

} else if (ch == 7) {

exit(0);

} else {

printf("Invalid Choice");

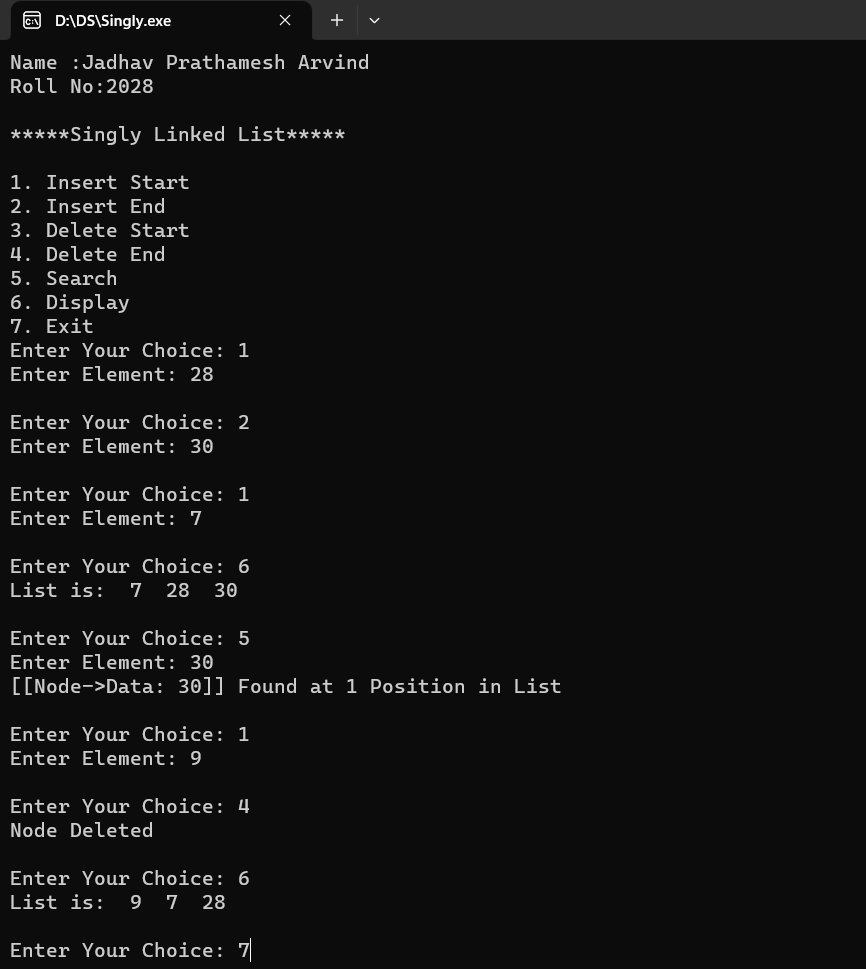
}

} while (ch != 7);

return 0;

}

* **Output :**

****

* **Result:**

Thus, the program for singly linked list was executed successfully.

**Ex. no.: 03 Implementation Of Doubly Linked List Date:**

**Aim:** To implement program for doubly linked list

**Algorithm:**

**1]** algorithm insertstart (datain)

1. allocate (newnode)
2. assign newnode data with datain
3. assign newnode next with head
4. assign newnode prev with null
5. if (head null)
6. assign head prev with newnode
7. end if
8. assign head with newnode

end insertstart

**2]** algorithm insertend (datain)

1. allocate (newnode)
2. assign newnode data with datain
3. assign newnode next with null
4. if (head null)
5. assign newnode prev with null
6. assign head with newnode
7. return
8. end if
9. assign temp with head
10. loop (temp next not null)
11. assign temp with temp next
12. end loop
13. assign temp next with newnode
14. assign newnode prev with temp

end insertend

**3]** algorithm deletestart ()

1. assign tempnode with head
2. if (head null)
3. return
4. end if
5. if (tempnode next null)
6. assign head with null
7. return
8. end if
9. assign head with head next
10. assign head prev with null
11. free (tempnode)

end deletestart

**4]** algorithm deleteend()

1. assign tempnode with head
2. if (head null)
3. return
4. end if
5. if (tempnode next null)
6. assign head with null
7. free (tempnode)
8. return
9. end if
10. loop (tempnode prev not null)
11. assign tempNode with tempNode next
12. end loop
13. assign tempNode prev next with NULL
14. free (tempNode)

end deleteEnd

**6]** Algorithm search (dataIn)

1. assign tempNode with head
2. assign count with 1 & flag with 0
3. if (tempNode equals NULL)
   * + 1. return
4. end if
5. loop (tempNode not equals NULL)
   * + 1. if (tempNode data equals data)

display node data and count

assign flag with 1

* + - 1. end if
      2. assign tempNode with tempNode next
      3. increment count

1. end loop
2. if (flag equals 0)
   * + 1. display data not found
3. end if

endsearch

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \* next;

struct Node \* prev;

};

struct Node \* head = NULL;

void insertStart(int data) {

struct Node \* newNode = (struct Node \* ) malloc(sizeof(struct Node));

newNode -> data = data;

newNode -> next = head;

newNode -> prev = NULL;

if (head != NULL)

head -> prev = newNode;

head = newNode;

}

void insertEnd(int data) {

struct Node \* newNode = (struct Node \* ) malloc(sizeof(struct Node));

newNode -> data = data;

newNode -> next = NULL;

if (head == NULL) {

head = newNode;

newNode -> prev = NULL;

return;

}

struct Node \* temp = head;

while (temp -> next != NULL)

temp = temp -> next;

temp -> next = newNode;

newNode -> prev = temp;

}

void deleteStart() {

struct Node \* tempNode = head;

if (head == NULL) {

printf("Linked List Empty\n");

return;

}

if (tempNode -> next == NULL) {

printf("Node Deleted\n");

head = NULL;

return;

}

head = head -> next;

head -> prev = NULL;

free(tempNode);

printf("Node Deleted\n");

}

void deleteEnd() {

struct Node \* tempNode = head;

if (head == NULL) {

printf("Linked List Empty\n");

return;

}

if (tempNode -> next == NULL) {

printf("Node Deleted\n");

head = NULL;

free(tempNode);

return;

}

while (tempNode -> next != NULL)

tempNode = tempNode -> next;

tempNode -> prev -> next = NULL;

free(tempNode);

printf("Node Deleted\n");

}

void display() {

struct Node \* end, \* node = head;

if (node == NULL) {

printf("List is Empty\n");

return;

}

printf("\nList in Forward direction: ");

while (node != NULL) {

printf(" %d ", node -> data);

end = node;

node = node -> next;

}

printf("\nList in backward direction: ");

while (end != NULL) {

printf(" %d ", end -> data);

end = end -> prev;

}

printf("\n");

}

void search(int data) {

struct Node \* node = head;

int count = 1, flag = 0;

if (node == NULL) {

printf("List is Empty\n");

} else {

while (node != NULL) {

if (node -> data == data) {

printf("[[Node->Data: %d]] Found at %d Position in List\n", node -> data, count);

flag = 1;

}

node = node -> next;

count++;

}

if (flag == 0) {

printf("Node not found");

}

printf("\n");

}

}

int getData() {

int data;

printf("Enter Element: ");

scanf("%d", & data);

return data;

}

int main() {

int ch;

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No : 2028\n\n”);

printf("\*\*\*\*\*Doubly Linked List\*\*\*\*\*\n\n");

printf("1. Insert Start\n2. Insert End\n3. Delete Start\n4. Delete End\n5. Search\n6. Display\n7. Exit");

do {

printf("\nEnter Your Choice: ");

scanf("%d", & ch);

if (ch == 1) {

insertStart(getData());

} else if (ch == 2) {

insertEnd(getData());

} else if (ch == 3) {

deleteStart();

} else if (ch == 4) {

deleteEnd();

} else if (ch == 5) {

search(getData());

} else if (ch == 6) {

display();

} else if (ch == 7) {

exit(0);

} else {

printf("Invalid Choice");

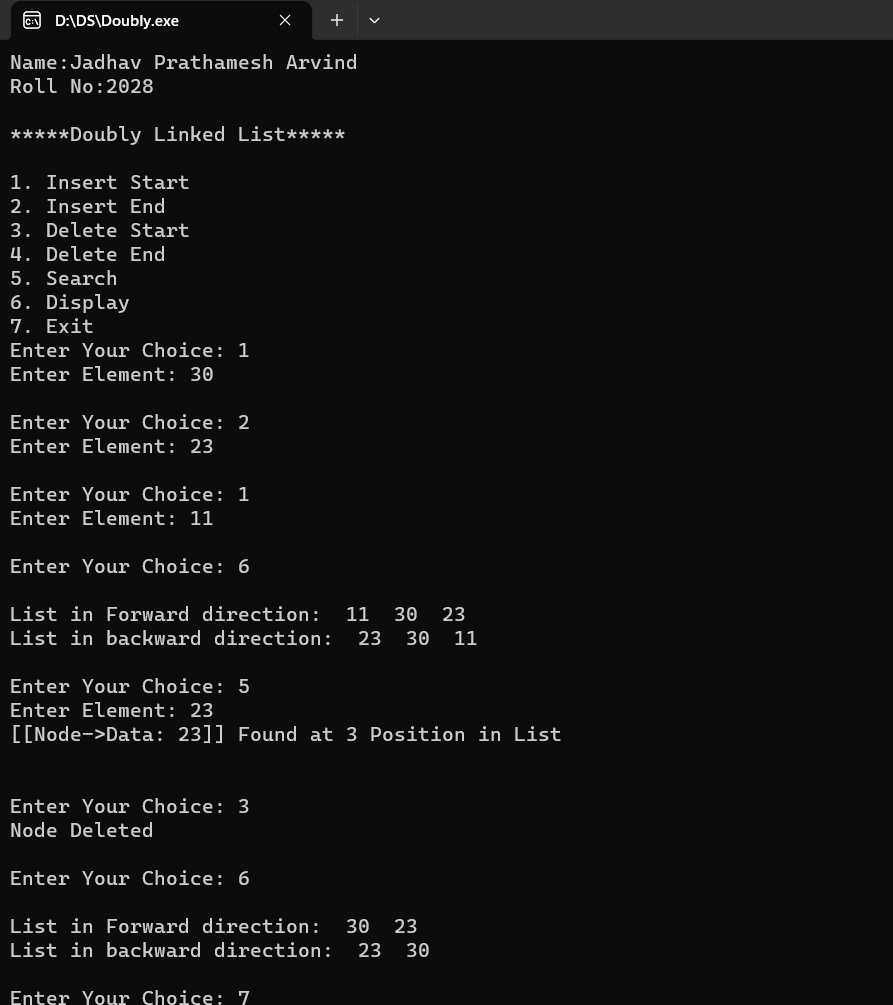
}

} while (ch != 7);

return 0;

}

* **Output :**

****

* **Result:**

Thus, the program for doubly linked list was executed successfully.

**Ex. No.: 04 4. A Implementation of Circular Singly Linked List Date:**

**Aim:** To Implement Program for Circular Singly Linked List

**Algorithm:**

**1]** Algorithm insertStart (dataIn)

1. assign temp with head
2. allocate (newNode)
3. assign newNode data with dataIn
4. assign newNode next with head
5. if (head not equals NULL)

loop (temp next not equals head)

1. assign temp with temp next

end loop

assign temp next with newNode

1. else

1. assign newNode next with newNode

1. assign head with newNode

end insertStart

**2]** Algorithm insertEnd (dataIn)

1. allocate (newNode)
2. assign newNode data with dataIn
3. assign newNode next with head
4. if (head equals NULL)
   * + 1. assign head with newNode
       2. assign newNode next with newNode
       3. return
5. end if
6. assign temp with head
7. loop (temp next not equals head)
   * + 1. assign temp with temp next
8. end loop
9. assign temp next with newNode

end insertEnd

**3]** Algorithm deleteStart ()

1. assign temp with head
2. assign prev with head
3. if (head equals NULL)
   * + 1. return
4. end if
5. loop (prev next not equals head)
   * + 1. assign prev with prev next
6. end loop
7. assign prev next with head next
8. assign head with head next
9. free(temp)

end deleteStart

**4]** Algorithm deleteEnd ()

1. assign temp with head
2. assign prev with NULL
3. if (head equals NULL)
   * + 1. return
4. end if
5. loop (temp next not equals head)
   * + 1. assign prev with temp
       2. assign temp with temp next
6. end loop
7. if (temp equals head)
   1. assign head with NULL
   2. free (temp)
8. else
   1. assign prev next with head
   2. free (temp)
9. end if

end deleteEnd

**5]** Algorithm search (dataIn)

1. assign temp with head
2. assign count with 1 & flag with 0
3. if (head not equals NULL)
   1. while (temp not equals head)
      1. if (temp data equals dataIn)
         1. display temp data & count
         2. assign flag with 1
      2. end if
   2. assign temp with temp next;
   3. increment count
   4. end loop
   5. if (flag equals 0)
      1. display data not found
   6. end if
4. else
   * + 1. display data not found
5. end if

end search

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \* next;

};

struct Node \* head = NULL;

void insertStart(int data) {

struct Node \* temp = head;

struct Node \* newNode = (struct Node \* ) malloc(sizeof(struct Node));

newNode -> data = data;

newNode -> next = head;

if (head != NULL) {

while (temp -> next != head)

temp = temp -> next;

temp -> next = newNode;

} else {

newNode -> next = newNode;

}

head = newNode;

}

void insertEnd(int data) {

struct Node \* newNode = (struct Node \* ) malloc(sizeof(struct Node));

newNode -> data = data;

newNode -> next = head;

if (head == NULL) {

head = newNode;

newNode -> next = newNode;

return;

}

struct Node \* temp = head;

while (temp -> next != head)

temp = temp -> next;

temp -> next = newNode;

}

void deleteStart() {

struct Node \* temp = head, \* prev = head;

if (head == NULL) {

printf("List is Empty\n");

return;

}

while (prev -> next != head)

prev = prev -> next;

prev -> next = head -> next;

head = head -> next;

free(temp);

printf("Node Deleted\n");

}

void deleteEnd() {

struct Node \* temp = head, \* prev;

if (head == NULL) {

printf("List is Empty\n");

return;

}

while (temp -> next != head) {

prev = temp;

temp = temp -> next;

}

if (temp == head) {

head = NULL;

free(temp);

} else {

prev -> next = head;

free(temp);

}

printf("Node Deleted\n");

}

void search(int data) {

struct Node \* temp = head;

int count = 1, flag = 0;

if (head != NULL) {

do {

if (temp -> data == data) {

printf("[[Node->Data: %d]] Found at %d Position in List", temp -> data, count);

flag = 1;

}

temp = temp -> next;

count++;

} while (temp != head);

if (flag == 0) {

printf("Node not found");

}

} else {

printf("List is Empty");

}

printf("\n");

}

void display() {

struct Node \* temp = head;

if (head != NULL) {

printf("List is: ");

do {

printf("%d ", temp -> data);

temp = temp -> next;

} while (temp != head);

} else {

printf("List is Empty");

}

printf("\n");

}

int getData() {

int data;

printf("Enter Element: ");

scanf("%d", & data);

return data;

}

int main() {

int ch;

printf(“Name: Jadhav Prathamesh Arvind\n”);

printf(“Roll No:2028\n\n”);

printf("\*\*\*\*\*Circular Singly Linked List\*\*\*\*\*\n\n");

printf("1. Insert Start\n2. Insert End\n3. Delete Start\n4. Delete End\n5. Search\n6. Display\n7. Exit");

do {

printf("\nEnter Your Choice: ");

scanf("%d", & ch);

if (ch == 1) {

insertStart(getData());

} else if (ch == 2) {

insertEnd(getData());

} else if (ch == 3) {

deleteStart();

} else if (ch == 4) {

deleteEnd();

} else if (ch == 5) {

search(getData());

} else if (ch == 6) {

display();

} else if (ch == 7) {

exit(0);

} else {

printf("Invalid Choice");

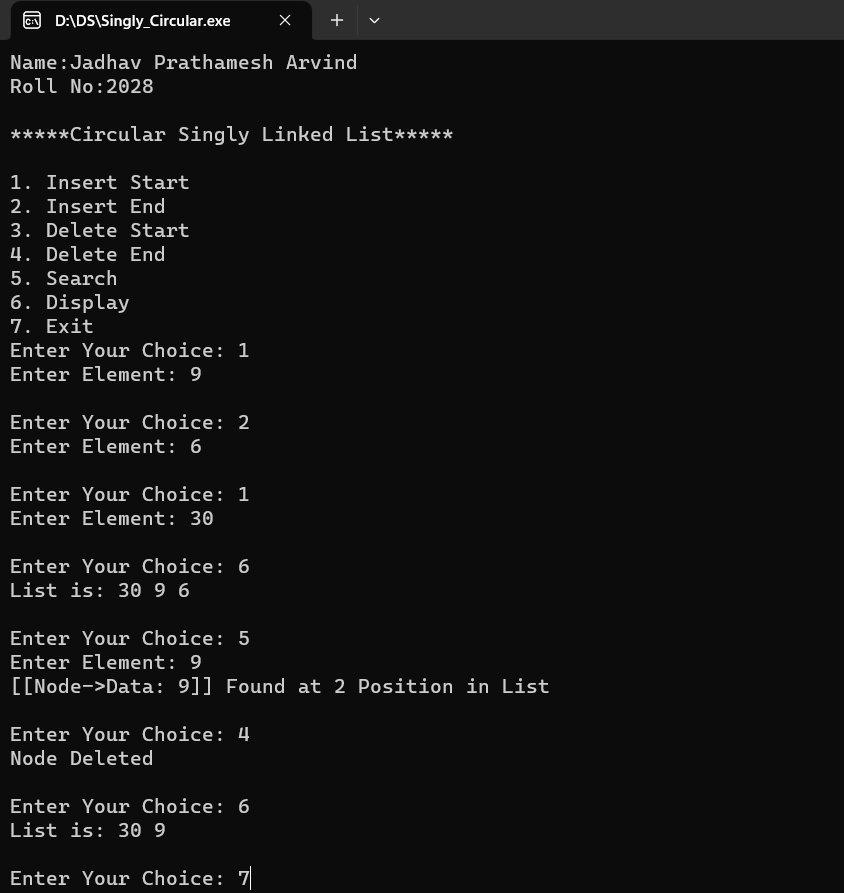
}

} while (ch != 7);

return 0;

}

* **Output :**

****

* **Result:**

Thus, the program of circular singly linked list was executed successfully.

**Ex. No.: 04 4.B Implementation of Circular Doubly Linked List Date:**

**Aim:** To Implement Program for Circular Singly Linked List

**Algorithm:**

**1]** Algorithm insertStart (dataIn)

1. allocate (new\_node)
2. if (head equals NULL)
   1. assign new\_node next with new\_node
   2. assign new\_node prev with new\_node
   3. assign head with new\_node
3. else
   1. assign head prev next with new\_node
   2. assign new\_node prev with head prev
   3. assign new\_node next with head
   4. assign head prev with new\_node
   5. assign head with new\_node
4. end if

end insertStart

**2]** Algorithm insertEnd (dataIn)

1. allocate (new\_node)
2. assign new\_node with dataIn
3. if (head equals NULL)
   1. assign new\_node next with new\_node
   2. assign new\_node prev with new\_node
   3. assign head with new\_node
4. else
   1. assign head prev next with new\_node
   2. assign new\_node prev with head prev
   3. assign new\_node next with head
   4. assign head prev with new\_node
5. end if

end insertEnd

**3]** Algorithm deleteStart ()

1. assign temp with head
2. if (temp equals NULL)
   1. return
3. end if
4. if (temp next equals temp)
   1. assign temp next with NULL
   2. assign temp prev with NULL
   3. free (temp)
   4. assign head with NULL
   5. return
5. end if
6. assign head with head next
7. assign head prev with temp prev
8. assign temp prev next with head
9. free (temp)

end deleteStart

**4]** Algorithm deleteEnd ()

1. if (head equals NULL)
   1. return
2. end if
3. assign last with head prev
4. assign secondToLast with last prev
5. if (last equals head)
   1. free (last)
   2. assign head with NULL
6. else
   1. assign secondToLast next with head
   2. assign head prev with secondToLast
   3. free (last)
7. end if

end deleteEnd

**5]** Algorithm search (dataIn) {

1. assign temp with head
2. assign count with 1 & flag with 0
3. if (head equals NULL)
   1. return
4. end if
5. loop (temp not equals head)
   1. if (temp data equals dataIn) {
      1. display temp data & count
      2. assign flag with 1
   2. end if
   3. assign temp with temp next
   4. increment count
6. end loop
7. if (flag equals 0) {
   1. display Data not found
8. end if

end search

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \* next;

struct Node \* prev;

};

struct Node \* head = NULL;

struct Node \* create(int data) {

struct Node \* new\_node = (struct Node \* ) malloc(sizeof(struct Node));

new\_node -> data = data;

new\_node -> next = NULL;

new\_node -> prev = NULL;

return new\_node;

}

void insertStart(int data) {

struct Node \* new\_node = create(data);

if (head == NULL) {

new\_node -> next = new\_node;

new\_node -> prev = new\_node;

head = new\_node;

} else {

head -> prev -> next = new\_node;

new\_node -> prev = head -> prev;

new\_node -> next = head;

head -> prev = new\_node;

head = new\_node;

}

}

void insertEnd(int data) {

struct Node \* new\_node = create(data);

if (head == NULL) {

new\_node -> next = new\_node;

new\_node -> prev = new\_node;

head = new\_node;

} else {

head -> prev -> next = new\_node;

new\_node -> prev = head -> prev;

new\_node -> next = head;

head -> prev = new\_node;

}

}

void deleteStart() {

struct Node \* temp = head;

if (temp == NULL) {

printf("List is Empty\n");

return;

}

if (temp -> next == temp) {

temp -> next = NULL;

temp -> prev = NULL;

free(temp);

head = NULL;

printf("Node Deleted\n");

return;

}

head = head -> next;

head -> prev = temp -> prev;

temp -> prev -> next = head;

free(temp);

printf("Node Deleted\n");

}

void deleteEnd() {

if (head == NULL) {

printf("List is Empty\n");

return;

}

struct Node \* last = head -> prev;

struct Node \* secondToLast = last -> prev;

if (last == head) {

free(last);

head = NULL;

} else {

secondToLast -> next = head;

head -> prev = secondToLast;

free(last);

}

printf("Node Deleted\n");

}

void display() {

struct Node \* temp = head;

if (head == NULL) {

printf("List is Empty\n");

return;

}

printf("\nList in Forward Direction: ");

do {

printf(" %d ", temp -> data);

temp = temp -> next;

} while (temp != head);

printf("\nList in Backward Direction: ");

do {

temp = temp -> prev;

printf(" %d ", temp -> data);

} while (temp != head);

printf("\n");

}

void search(int data) {

struct Node \* temp = head;

int count = 1, flag = 0;

if (head == NULL) {

printf("List is Empty\n");

return;

}

do {

if (temp -> data == data) {

printf("[[Node->Data: %d]] Found at %d Position in List\n", temp -> data, count);

flag = 1;

}

temp = temp -> next;

count++;

} while (temp != head);

if (flag == 0)

printf("Node not found");

}

int getData() {

int data;

printf("Enter Element: ");

scanf("%d", & data);

return data;

}

int main() {

int ch;

printf(“Name : Jadhav Prathmesh Arvind\n”);

printf(“Roll No : 2028\n\n”);

printf("\*\*\*\*\*Circular Doubly Linked List\*\*\*\*\*\n\n");

printf("1. Insert Start\n2. Insert End\n3. Delete Start\n4. Delete End\n5. Search\n6. Display\n7. Exit");

do {

printf("\nEnter Your Choice: ");

scanf("%d", & ch);

if (ch == 1) {

insertStart(getData());

} else if (ch == 2) {

insertEnd(getData());

} else if (ch == 3) {

deleteStart();

} else if (ch == 4) {

deleteEnd();

} else if (ch == 5) {

search(getData());

} else if (ch == 6) {

display(head);

} else if (ch == 7) {

exit(0);

} else {

printf("Invalid Choice");

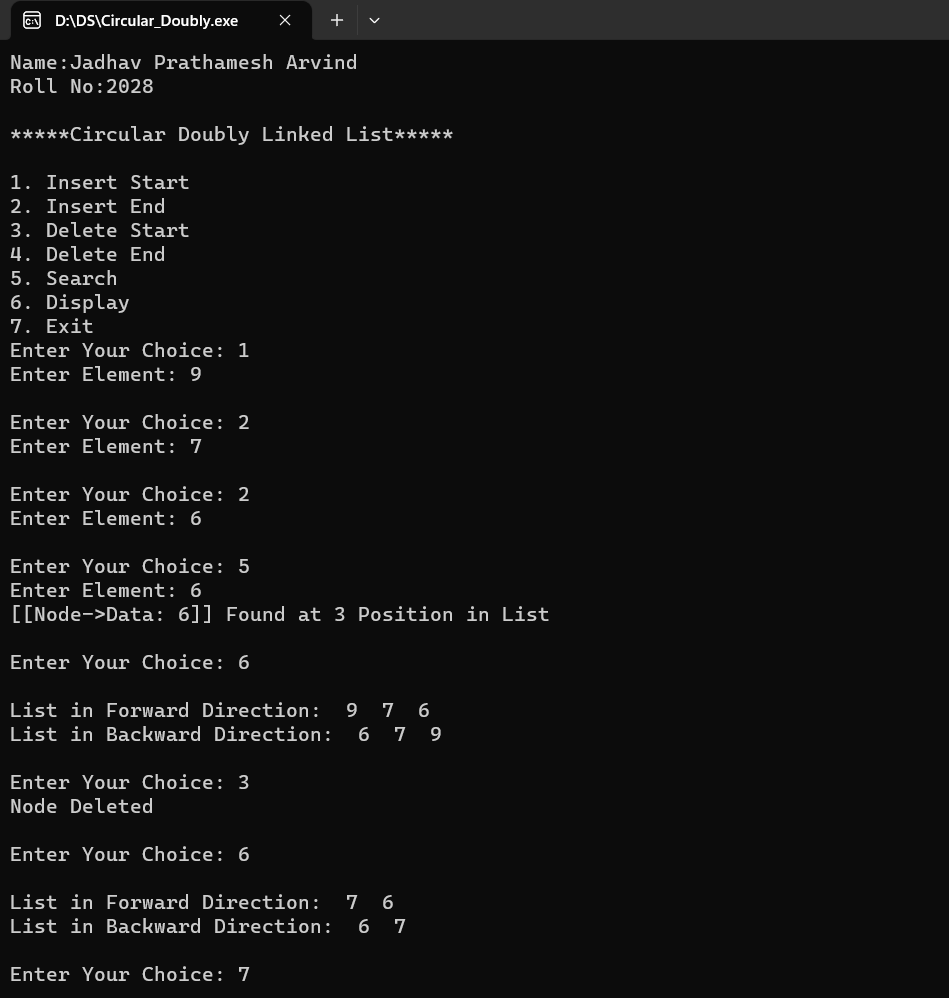
}

} while (ch != 7);

return 0;

}

* **Output :**

****

* **Result:**

Thus, the program of circular doubly linked list was executed successfully.

**Ex. No.: 05 5.A Implementation of Stack using Array Date:**

**Aim:** To Implement Program for Stack using Array

**Algorithm:**

**1]** Algorithm push ()

1. assign data with 0
2. if (top equals MAX\_SIZE - 1)
   * 1. display Overflow State
     2. return
3. end if
4. data = get input from user
5. increment top
6. assign stack[top] with data

end push

**2]** Algorithm pop ()

1. if (top equals -1)
   1. display Underflow State
   2. return
2. endif
3. assign value with stack[top]
4. decrement top
5. display value

end pop

**3]** Algorithm peek ()

1. if (top not equals -1)
   1. display stack[top]
2. else
   1. display Underflow State
3. endif

end pop

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 100

int stack[MAX\_SIZE];

int top = -1;

void isEmpty() {

if (top == -1) {

printf("Stack is empty: Underflow State\n");

} else {

printf("Stack is not empty\n");

}

}

void push() {

int data;

if (top == MAX\_SIZE - 1) {

printf("Error: stack is full\n");

return;

}

printf("Enter Value To Push: ");

scanf("%d", & data);

top++;

stack[top] = data;

}

void pop() {

if (top == -1) {

printf("Error: Stack is Empty\n");

return;

}

int value = stack[top];

top--;

printf("Popped Value: %d\n", value);

}

void peek() {

if (top != -1)

printf("Topmost Element: %d\n", stack[top]);

else

printf("Stack is Empty: Underflow State\n");

}

int main() {

int ch;

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No : 2028\n\n”);

printf("\n1. Push\n2. Pop\n3. Peek\n4. is Empty?\n5. Exit");

do {

printf("\nEnter Your Choice: ");

scanf("%d", & ch);

if (ch == 1) {

push();

} else if (ch == 2) {

pop();

} else if (ch == 3) {

peek();

} else if (ch == 4) {

isEmpty();

} else if (ch == 5) {

exit(0);

} else {

printf("Invalid Choice\n");

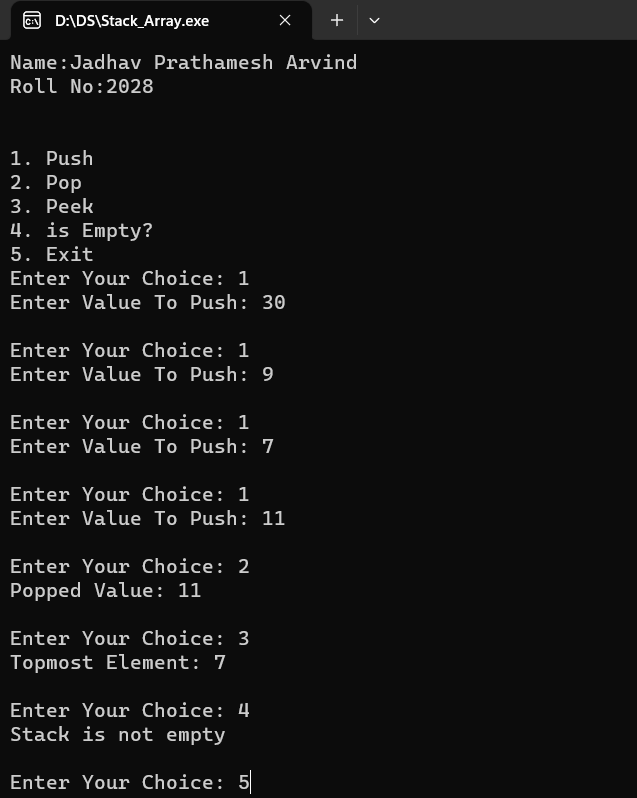
}

} while (ch != 5);

return 0;

}

* **Output :**

****

* **Result :**

Thus, the program of stack using array was executed successfully.

**Ex. No.: 05 5.B Implementation of Stack using List Date:**

**Aim:** To Implement Program for Stack using List

**Algorithm:**

**1]** Algorithm push ()

1. allocate (ptr)
2. assign data with 0
3. get input from user and store in data
4. if (head equals NULL)
   1. assign ptr val with data
   2. assign ptr next with NULL
   3. assign head with ptr
5. else
   1. assign ptr val with data
   2. assign ptr next with head
   3. assign head with ptr
6. end if

end push

**2]** Algorithm pop ()

1. assign item with 0
2. assign ptr with NULL
3. if (head equals NULL)
   1. display Underflow State
4. else
   1. assign item with head val
   2. assign ptr with head
   3. assign head with head next
   4. free (ptr)
   5. display item
5. end if

end pop

**3]** Algorithm peek ()

1. if (head not equals NULL)
   1. display head val
2. else
   1. display Underflow State
3. end if

end peek

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#include <stdlib.h>

struct Node {

int val;

struct Node \* next;

} \* head;

void isEmpty() {

if (head == NULL) {

printf("Stack is empty: Underflow State\n");

} else {

printf("Stack is not empty\n");

}

}

void push() {

struct Node \*ptr = (struct Node \*) malloc(sizeof(struct Node));

int data;

printf("Enter Value To Push: ");

scanf("%d", & data);

if (head == NULL) {

ptr -> val = data;

ptr -> next = NULL;

head = ptr;

} else {

ptr -> val = data;

ptr -> next = head;

head = ptr;

}

}

void pop() {

int item;

struct Node \* ptr;

if (head == NULL) {

printf("Underflow State: can't remove any item\n");

} else {

item = head -> val;

ptr = head;

head = head -> next;

free(ptr);

printf("Popped Element: %d\n", item);

}

}

void peek() {

if (head != NULL)

printf("Topmost Element: %d\n", head -> val);

else

printf("Stack is empty: Underflow State\n");

}

int main() {

int ch;

printf("Name: Jadhav Prathamesh Arvind\n");

printf("Roll No:2028\n\n");

printf("1. Push\n2. Pop\n3. Peek\n4. is Empty?\n5. Exit");

do {

printf("\nEnter Your Choice: ");

scanf("%d", & ch);

if (ch == 1) {

push();

} else if (ch == 2) {

pop();

} else if (ch == 3) {

peek();

} else if (ch == 4) {

isEmpty();

} else if (ch == 5) {

exit(0);

} else {

printf("Invalid Choice\n");

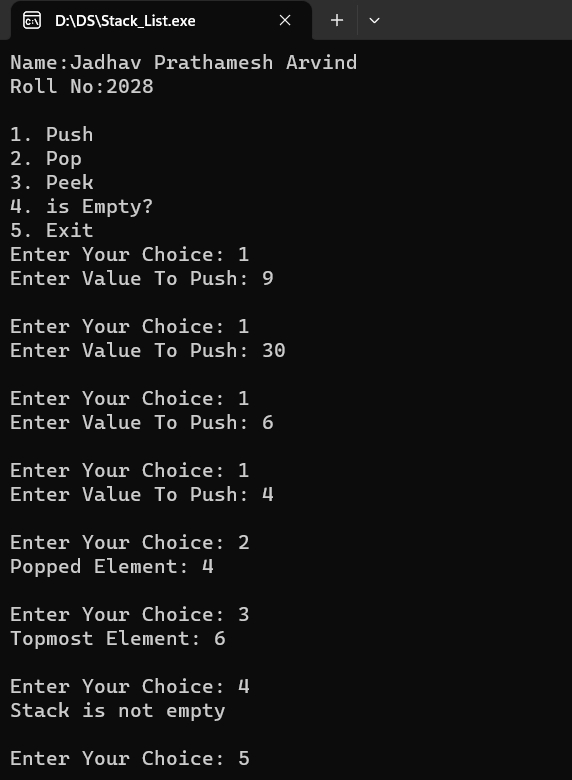
}

} while (ch != 7);

return 0;

}

* **Output :**

s

* **Result:**

Thus, the program of stack using list was executed successfully.

**Ex. No.: 06 6.A Implementation of Queue using Array Date:**

**Aim:** To Implement Program for Stack using List

**Algorithm:**

**1]** Algorithm enqueue ()

1. assign data with 0
2. if ((rear + 1) % MAX\_SIZE equals front)
   1. return
3. end if
4. get input user & store in data
5. if (front equals -1 && rear equals -1) {
   1. assign front with rear with 0
6. else
   1. assign rear with (rear + 1) % MAX\_SIZE
7. end if
8. assign queue[rear] with data

end enqueue

2] Algorithm dequeue () {

1. assign value with queue[front]
2. if (front equals -1 && rear equals -1)
   1. return
3. end if
4. if (front equals rear)
   1. assign front & rear with -1
5. else
   1. assign front with (front + 1) % MAX\_SIZE
6. end if
7. display value

end dequeue

* **Program:**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#define MAX\_SIZE 10

int queue[MAX\_SIZE];

int front = -1, rear = -1;

void enqueue() {

int data;

if ((rear + 1) % MAX\_SIZE == front) {

printf("Queue is full\n");

return;

}

printf("Enter Value To Enqueue: ");

scanf("%d", & data);

if (front == -1 && rear == -1) {

front = rear = 0;

} else {

rear = (rear + 1) % MAX\_SIZE;

}

queue[rear] = data;

}

void dequeue() {

int value = queue[front];

if (front == -1 && rear == -1) {

printf("Queue is empty\n");

return;

}

if (front == rear) {

front = rear = -1;

} else {

front = (front + 1) % MAX\_SIZE;

}

printf("Dequeued Element: %d\n", value);

}

void display() {

int i;

if (front == -1)

printf("Queue is empty\n");

else {

printf("Queue is : ");

for (i = front; i <= rear; i++)

printf("%d ", queue[i]);

printf("\n");

}

}

int main() {

int ch;

printf("Name: Jadhav Prathamesh Arvind \n");

printf("Roll No:2028\n\n");

printf("\*\*Static Queue\*\*\n\n");

printf("1. Enqueue\n2. Dequeue\n3. Display All Elements\n4. Exit");

do {

printf("\nEnter Your Choice: ");

scanf("%d", & ch);

if (ch == 1) {

enqueue();

} else if (ch == 2) {

dequeue();

} else if (ch == 3) {

display();

} else if (ch == 4) {

exit(0);

} else {

printf("Invalid Choice\n");

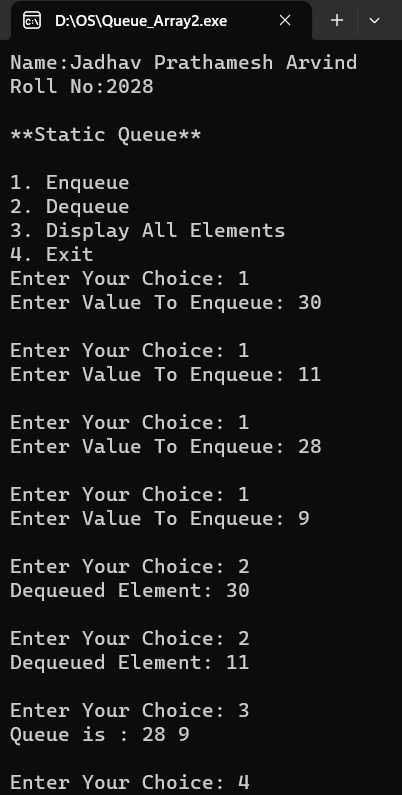
}

} while (ch != 4);

return 0;

}

* **Output:**

****

* **Result:**

Thus**,** the program of queue using array was executed successfully.

**Ex. No.: 06 6.B Implementation of Queue using List Date:**

**Aim:** To Implement Program for Queue using List

**Algorithm:**

**1]** Algorithm enqueue ()

1. assign data with 0
2. get input user & store in data
3. allocate (nptr)
4. assign nptr data with data
5. assign nptr next with NULL
6. if (rear equals NULL)
   1. assign front with nptr
   2. assign rear with nptr
7. else
   1. assign rear next with nptr
   2. assign rear with rear next
8. end if

end enqueue

**2]** Algorithmdequeue ()

1. if (front not equals NULL)
   1. assign temp with front
   2. assign front with front next
2. if (front equals NULL)
   1. assign rear with NULL
3. end if
4. display temp data
5. free (temp)
6. end if

end dequeue

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*next;

};

struct Node \*front = NULL, \*rear = NULL;

void enqueue() {

int data;

printf("Enter Value To Enqueue: ");

scanf("%d", & data);

struct Node \* nptr = malloc(sizeof(struct Node));

nptr -> data = data;

nptr -> next = NULL;

if (rear == NULL) {

front = nptr;

rear = nptr;

} else {

rear -> next = nptr;

rear = rear -> next;

}

}

void dequeue() {

if (front == NULL) {

printf("Queue is Empty\n");

} else {

struct Node \* temp = front;

front = front -> next;

if (front == NULL) {

rear = NULL;

}

printf("Dequeued Element: %d\n", temp -> data);

free(temp);

}

}

void display() {

struct Node \* temp = front;

if (front == NULL) {

printf("Queue is Empty\n");

return;

}

printf("Queue is: ");

while (temp != NULL) {

printf(" %d ", temp -> data);

temp = temp -> next;

}

printf("\n");

}

int main() {

int ch;

printf("Name: Jadhav Prathamesh Arvind \n");

printf("Roll No:2028\n\n");

printf("\*\*Dynamic Queue\*\*\n\n");

printf("1. Enqueue\n2. Dequeue\n3. Display All Elements\n4. Exit");

do {

printf("\nEnter Your Choice: ");

scanf("%d", &ch);

if (ch == 1) {

enqueue();

} else if (ch == 2) {

dequeue();

} else if (ch == 3) {

display();

} else if (ch == 4) {

exit(0);

} else {

printf("Invalid Choice\n");

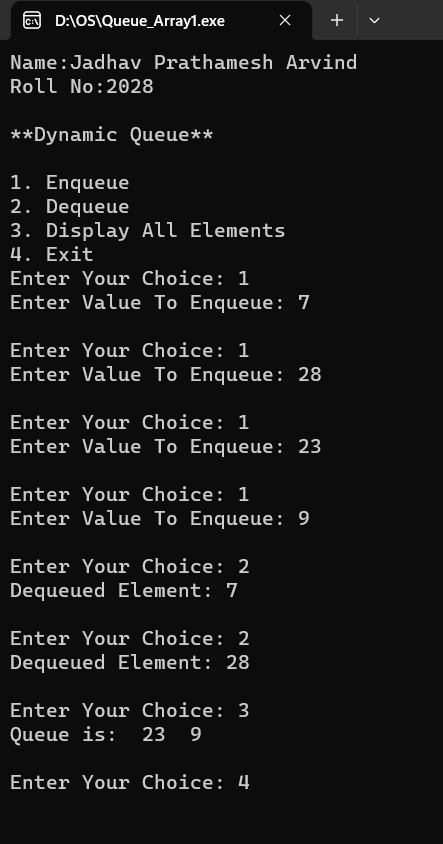
}

} while(ch != 4);

return 0;

}

* **Output :**

****

* **Result:**

Thus the program of queue using list was executed successfully.

**Ex. No.: 07 7.A Implementation of Infix to Postfix Expression Date:**

**Aim:** To Implement Program of Infix to Postfix Conversion of an Expression

**Algorithm:**

**1]** Algorithm push(item, stack[], \*top)

1. assign stack[++(\*top)] with item;

end push

**2]** Algorithm pop(stack[], \*top)

1. return stack[(\*top)--]

end pop

**3]** Algorithm is\_operator(symbol)

1. if (symbol equals ('+' || '-' || '\*' || '/' || '^'))
   1. return 1
2. end if
3. return 0

end is\_operator

1. Declare the variables infix[MAX\_EXPR\_SIZE], postfix[MAX\_EXPR\_SIZE], and stack[MAX\_EXPR\_SIZE] of type char and variable top of type int.
2. Print "Infix To Postfix" and "Enter an Infix Expression: ".
3. Read the input infix expression into the infix variable using scanf().
4. For each character in the infix expression from left to right, do the following:

* If the character is an alphanumeric character, append it to the postfix expression.
* Else if the character is an opening parenthesis '(', push it onto the stack.
* Else if the character is a closing parenthesis ')', pop elements from the stack and append them to the postfix expression until the corresponding opening parenthesis is found in the stack, and then pop it from the stack.
* Else if the character is an operator (+, -, \*, /, ^), pop operators from the stack and append them to the postfix expression until an operator with lower precedence or an opening parenthesis is found in the stack, and then push the current operator onto the stack.

1. After processing all characters in the infix expression, pop any remaining operators from the stack and append them to the postfix expression.
2. Terminate the postfix expression with a null character.
3. Print the postfix expression.

* **Program:**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define MAX\_EXPR\_SIZE 100

void push(char item, char stack[], int \*top) {

stack[++(\*top)] = item;

}

char pop(char stack[], int \*top) {

return stack[(\*top)--];

}

int is\_operator(char symbol) {

if (symbol == '+' || symbol == '-' || symbol == '\*' || symbol == '/' || symbol == '^') {

return 1;

}

return 0;

}

int precedence(char symbol) {

if (symbol == '^') {

return 3;

} else if (symbol == '\*' || symbol == '/') {

return 2;

} else if (symbol == '+' || symbol == '-') {

return 1;

} else {

return 0;

}

}

int main() {

char infix[MAX\_EXPR\_SIZE], postfix[MAX\_EXPR\_SIZE];

char stack[MAX\_EXPR\_SIZE];

int top = 0, i, j;

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No : 2028\n\n”);

printf("\*\*Infix To Postfix\*\*\n\n");

printf("Enter an Infix Expression: ");

scanf("%s", infix);

for (i = 0, j = 0; infix[i] != '\0'; i++) {

if (isalnum(infix[i])) {

postfix[j++] = infix[i];

} else if (infix[i] == '(') {

push(infix[i], stack, &top);

} else if (infix[i] == ')') {

while (stack[top] != '(') {

postfix[j++] = pop(stack, &top);

}

pop(stack, &top);

} else if (is\_operator(infix[i])) {

while (is\_operator(stack[top]) && precedence(infix[i]) <= precedence(stack[top])) {

postfix[j++] = pop(stack, &top);

}

push(infix[i], stack, &top);

}

}

while (top > 0) {

postfix[j++] = pop(stack, &top);

}

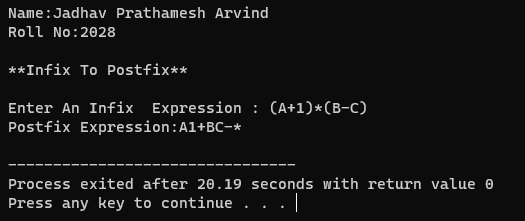
postfix[j] = '\0';

printf("Postfix expression: %s\n", postfix);

return 0;

}

* **Output :**



* **Result:**

Thus, the program of infix to postfix conversion was studied successfully.

**Ex. No.: 07 7.B Implementation of Circular Queue using Array Date:**

**Aim:** To Implement Program of Circular Queue using Array

**Algorithm:**

**1]** Algorithm enqueue(Queue, value)

1. if (is\_full(Queue))
   1. return;
2. end if
3. assign Queue data[Queue -> rear] with value
4. assign Queue rear with (Queue -> rear + 1) % QUEUE\_SIZE

end enqueue

**2]** Algorithm void dequeue(Queue)

1. if (is\_empty(Queue))
   1. return
2. end if
3. assign value with Queue data[Queue front];
4. assign Queue front with (Queue front + 1) % QUEUE\_SIZE;
5. display value

end dequeue

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#include <stdlib.h>

#define QUEUE\_SIZE 5

typedef struct {

int data[QUEUE\_SIZE];

int front;

int rear;

} Queue;

void enqueue(Queue \* q, int value) {

if (is\_full(q)) {

printf("Queue is full\n");

return;

}

q -> data[q -> rear] = value;

q -> rear = (q -> rear + 1) % QUEUE\_SIZE;

}

void dequeue(Queue \* q) {

if (is\_empty(q)) {

printf("Queue is empty\n");

return;

}

int value = q -> data[q -> front];

q -> front = (q -> front + 1) % QUEUE\_SIZE;

printf("Dequeued: %d\n", value);

}

int is\_full(Queue \* q) {

return (q -> rear + 1) % QUEUE\_SIZE == q -> front;

}

int is\_empty(Queue \* q) {

return q -> front == q -> rear;

}

void display(Queue \* q) {

int i;

if (is\_empty(q)) {

printf("Queue is empty\n");

return;

} else {

printf("Queue: ");

for (i = q -> front; i != q -> rear; i = (i + 1) % QUEUE\_SIZE) {

printf("%d ", q -> data[i]);

}

}

printf("\n");

}

int main() {

int ch, data;

Queue q;

q.front = 0;

q.rear = 0;

printf("Name : Jadhav Prathamesh Arvind\n");

printf("Roll No: 2028\n\n");

printf("Circular Queue\n\n");

printf("1. Enqueue\n2. Dequeue\n3. Display All Elements\n4. Exit");

do {

printf("\nEnter Your Choice: ");

scanf("%d", & ch);

if (ch == 1) {

printf("Enter Element: ");a

scanf("%d", & data);

enqueue( & q, data);

} else if (ch == 2) {

dequeue( & q);

} else if (ch == 3) {

display( & q);

} else if (ch == 4) {

exit(0);

} else {

printf("Invalid Choice\n");

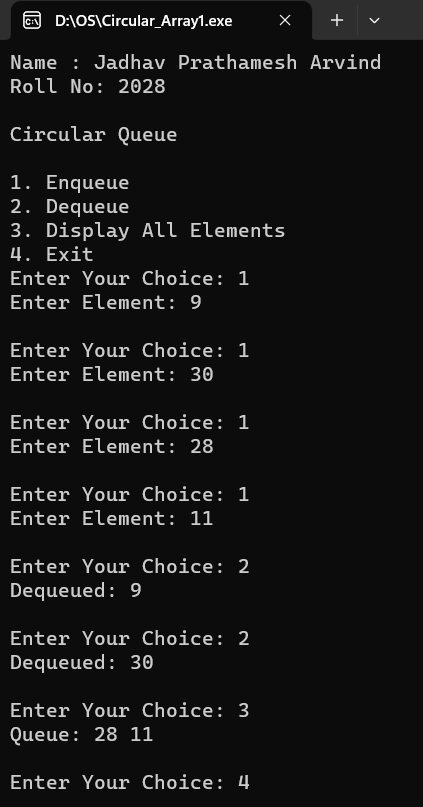
}

} while (ch != 4);

return 0;

}

* **Output :**

****

* **Result:**

Thus, the program of circular queue using array was executed succesfully.

**Ex. No.: 07 7.C Implementation of Double Ended Queue Date:**

**Aim:** To Implement Program of Double Ended Queue using Array

**Algorithm:**

**1]** Algorithm insert\_front(deque, val)

1. if (is\_full(deque))
   1. return
2. else if (is\_empty(deque))
   1. assign q front with 0
   2. assign q rear with 0
3. else if (q front equals 0)
   1. assign q front with MAX\_SIZE - 1
4. else
   1. decrement q front
5. end if
6. assign q items[q front] with val

end insert\_front

**2]** Algorithm insert\_rear(deque, val)

1. if (is\_full(deque))
   1. return;
2. else if (is\_empty(deque))
   1. assign q front with 0
   2. assign q rear with 0
3. else if (q rear equals MAX\_SIZE - 1)
   1. assign q rear with 0
4. else
   1. increment q rear
5. end if
6. assign q items[q rear] with val

end insert\_rear

**3]** Algorithm delete\_front(deque)

1. if (is\_empty(deque))
   1. return;
2. else if (q front equals q rear)
   1. assign q front with -1
   2. assign q rear with -1
3. else if (q front equals MAX\_SIZE - 1)
   1. assign q front with 0
4. else
   1. increment q front
5. end if

end deque

**4]** Algorithm delete\_rear(deque)

1. if (is\_empty(deque))
   1. return
2. else if (q front equals q rear)
   1. assign q front with -1
   2. assign q rear with -1
3. else if (q rear equals 0)
   1. assign q rear with MAX\_SIZE - 1;
4. else
   1. decrement q rear
5. end if

end delete\_rear

* **Program:**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 10

typedef struct deque {

int items[MAX\_SIZE];

int front;

int rear;

} deque;

void init(deque \* q) {

q -> front = -1;

q -> rear = -1;

}

int is\_empty(deque \* q) {

return (q -> front == -1 && q -> rear == -1);

}

int is\_full(deque \* q) {

return (q -> rear + 1) % MAX\_SIZE == q -> front;

}

void insert\_front(deque \* q, int val) {

if (is\_full(q)) {

printf("Queue is full\n");

return;

} else if (is\_empty(q)) {

q -> front = 0;

q -> rear = 0;

} else if (q -> front == 0) {

q -> front = MAX\_SIZE - 1;

} else {

q -> front--;

}

q -> items[q -> front] = val;

}

void insert\_rear(deque \* q, int val) {

if (is\_full(q)) {

printf("Queue is full\n");

return;

} else if (is\_empty(q)) {

q -> front = 0;

q -> rear = 0;

} else if (q -> rear == MAX\_SIZE - 1) {

q -> rear = 0;

} else {

q -> rear++;

}

q -> items[q -> rear] = val;

}

void delete\_front(deque \* q) {

if (is\_empty(q)) {

printf("Queue is empty\n");

return;

} else if (q -> front == q -> rear) {

q -> front = -1;

q -> rear = -1;

} else if (q -> front == MAX\_SIZE - 1) {

q -> front = 0;

} else {

q -> front++;

}

printf("Element Dequed\n");

}

void delete\_rear(deque \* q) {

if (is\_empty(q)) {

printf("Queue is empty\n");

return;

} else if (q -> front == q -> rear) {

q -> front = -1;

q -> rear = -1;

} else if (q -> rear == 0) {

q -> rear = MAX\_SIZE - 1;

} else {

q -> rear--;

}

printf("Element Dequed\n");

}

void display(deque \* q) {

if (is\_empty(q)) {

printf("Queue is empty\n");

return;

}

int i;

for (i = q -> front; i != q -> rear; i = (i + 1) % MAX\_SIZE) {

printf("%d ", q -> items[i]);

}

printf("%d\n", q -> items[i]);

}

int main() {

int ch, data;

deque q;

init( & q);

printf("Name: Jadhav Prathamesh Arvind\n");

printf("Roll No: 2028\n\n");

printf("\*\*Double Ended Queue\*\*\n\n");

printf("1. Enqueue Start\n2. Enqueue End\n3. Dequeue Start\n4. Dequeue End\n5. Display\n6. Exit");

do {

printf("\nEnter Your Choice: ");

scanf("%d", & ch);

switch (ch) {

case 1:

printf("Enter Element: ");

scanf("%d", & data);

insert\_front( & q, data);

break;

case 2:

printf("Enter Element: ");

scanf("%d", & data);

insert\_rear( & q, data);

break;

case 3:

delete\_front( & q);

break;

case 4:

delete\_rear( & q);

break;

case 5:

display( & q);

break;

case 6:

exit(0);

break;

default:

printf("Invalid Choice\n");

break;

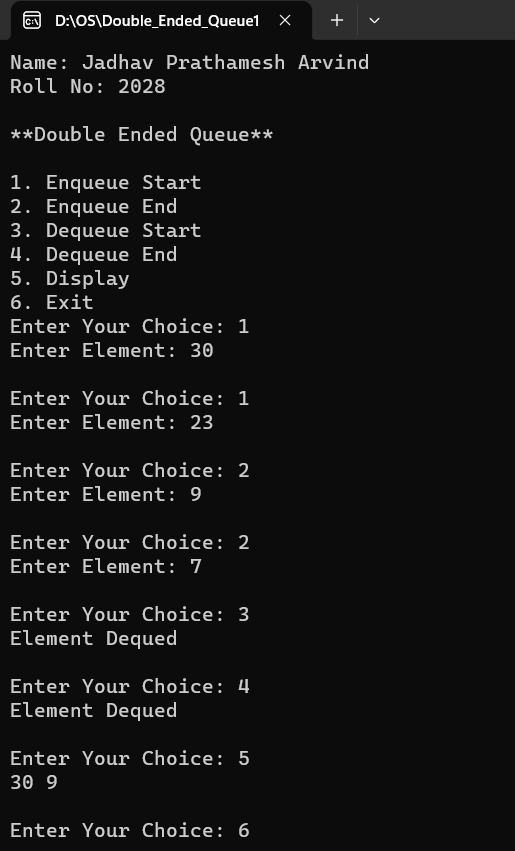
}

} while (ch != 6);

return 0;

}

* **Output :**

****

* **Result:**

Thus**,**  the program of linear search in array was executed successfully.

**Ex. No.: 08 8.A Implementation of Linear Search Date:**

**Aim:** To Implement Program of Linear Search in Array

**Algorithm:**

1] Algorithm linear\_search(array, n, key)

1. assign i with 0
2. loop (i < n)
   1. if (array[i] equals key)
      1. assign pos[ns] with i
      2. increment ns
   2. end if
   3. incremnt i
3. end loop

end linear\_search

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#define MAX 50

int arr[MAX], pos[MAX], ns = 0;

void linear\_search(int \* array, int n, int key) {

int i;

for (i = 0; i < n; i++) {

if (array[i] == key) {

pos[ns] = i;

ns++;

}

}

}

void insert(int \* array, int sizearr) {

int i;

for (i = 0; i < sizearr; i++) {

printf("Arr[%d]: ", i);

scanf("%d", & array[i]);

}

}

void display(int \* array, int n) {

int i;

for (i = 0; i < n; i++) {

if (i == n - 1) {

printf("%d ", array[i]);

} else {

printf("%d, ", array[i]);

}

}

}

int main() {

int sizearr, data, i, n;

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No : 2028\n\n”);

printf("====Linear Search (Array)====\n\n");

printf("Enter Size Of Array: ");

do {

scanf("%d", & sizearr);

if (sizearr <= 0) {

printf("(Invalid) Enter a Non-Zero Number: ");

}

} while (sizearr <= 0);

printf("\nEnter Data For Following Elements: \n");

insert(arr, sizearr);

printf("\nEntered Array = ");

display(arr, sizearr);

printf("\n\nEnter Data To Search: ");

scanf("%d", & n);

linear\_search(arr, sizearr, n);

if (ns != 0) {

printf("\nData found at index ");

display(pos, ns);

} else {

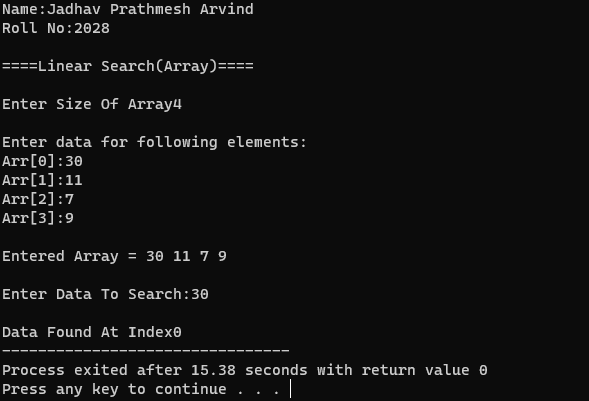
printf("\nData not found\n");

}

return 0;

}

* **Output :**



* **Result:**

Thus**,**  the program of linear search in array was executed successfully.

**Ex. No.: 08 8.B Implementation of Binary Search Date:**

**Aim:** To Implement Program of Binary Search in Array

**Algorithm:**

**1]** Algorithm sort (array, size)

1. assign i, j and temp with 0
2. loop (i < size)
   1. assign j with i + 1
   2. loop (j < size) {
      1. if (array[j] < array[i])
3. assign temp with array[i]
4. assign array[i] with array[j]
5. assign array[j] with temp
   * 1. end if
     2. increment j
   1. end loop
   2. increment i
6. end loop

end sort

2] Algorithm binary\_search (array, first, lastidx, x)

1. assign i with 1
2. if (first > lastidx)
   1. return -1
3. end if
4. assign mid with first + (lastidx - first) / 2
5. if (array[mid] equals x)
   1. return mid
6. else if (array[mid] > x)
   1. return binary\_search(array, first, mid - 1, x)
7. else
   1. return binary\_search(array, mid + 1, lastidx, x)
8. end if
9. return -1

end binary\_search

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#define MAX 50

int arr[MAX], pos[MAX], ns = 0;

void sort(int \* array, int size) {

int i, j, temp;

for (i = 0; i < size; i++) {

for (j = i + 1; j < size; j++) {

if (array[j] < array[i]) {

temp = array[i];

array[i] = array[j];

array[j] = temp;

}

}

}

}

int binary\_search(int \* array, int first, int lastidx, int x) {

int i = 1;

if (first > lastidx) {

return -1;

}

int mid = first + (lastidx - first) / 2;

if (array[mid] == x) {

return mid;

} else if (array[mid] > x) {

return binary\_search(array, first, mid - 1, x);

} else {

return binary\_search(array, mid + 1, lastidx, x);

}

return -1;

}

void insert(int \* array, int sizearr) {

int i;

for (i = 0; i < sizearr; i++) {

printf("Arr[%d]: ", i);

scanf("%d", & array[i]);

}

}

void display(int \* array, int n) {

int i;

for (i = 0; i < n; i++) {

if (i == n - 1) {

printf("%d ", array[i]);

} else {

printf("%d, ", array[i]);

}

}

}

int main() {

int sizearr, data, i, n;

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No : 2028\n\n”);

printf("====Binary Search====\n\n");

printf("Enter Size Of Array: ");

do {

scanf("%d", & sizearr);

if (sizearr <= 0) {

printf("(Invalid) Enter a Non-Zero Number: ");

}

} while (sizearr <= 0);

printf("\nEnter Data For Following Elements: \n");

insert(arr, sizearr);

printf("\nEntered Array = ");

display(arr, sizearr);

printf("\nSorted Array = ");

sort(arr, sizearr);

display(arr, sizearr);

printf("\n\nEnter Data To Search: ");

scanf("%d", & n);

int idx = binary\_search(arr, 0, sizearr - 1, n);

if (idx != -1) {

printf("\nData found at index %d", idx);

} else {

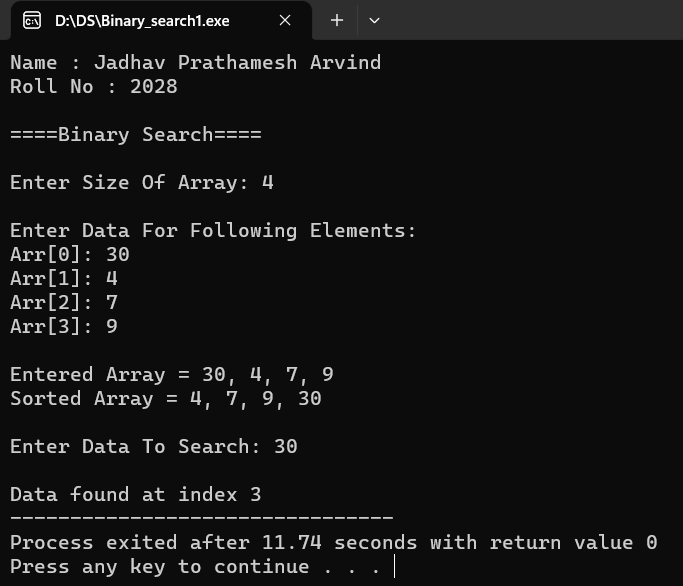
printf("\nData not found\n");

}

return 0;

}

* **Output :**

****

* **Result:**

Thus the program for binray search in array was executed successfully.

**Ex. No.: 09 9.A Implementation of Bubble Sort** **Date:**

**Aim:** To Implement Program of Bubble Sort

**Algorithm:**

**1]** Algorithm bubbleSort(array[], size)

1. assign I & temp with 0
2. assign step with size - 1
3. loop (step > 0)
   1. loop (i < step)
4. if (array[i] > array[i + 1])
   * 1. swap(array[i], array[i+1])
5. end if
6. increment i
   1. end loop
   2. decrement step
7. end loop

end bubbleSort

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

void swap(int \* xp, int \* yp) {

int temp = \* xp;

\* xp = \* yp;

\* yp = temp;

}

void bubbleSort(int array[], int size) {

int i, step, temp;

for (step = size - 1; step > 0; step--) {

for (i = 0; i < step; i++) {

if (array[i] > array[i + 1]) {

swap(&array[i], &array[i+1]);

}

}

}

}

void display(int \* arr, int size) {

int i;

for (i = 0; i < size; i++)

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

}

void insert(int \* arr, int size) {

int i;

printf("\nEnter Data For Following Elements\n");

for (i = 0; i < size; i++) {

printf("Arr[%d]: ", i);

scanf("%d", & arr[i]);

}

}

int main() {

int arr[50], size\_arr;

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No : 2028\n\n”);

printf("=== Bubble Sort ===\n\n");

printf("Enter Size Of Array: ");

do {

scanf("%d", &size\_arr);

if (size\_arr <= 0) {

printf("(Invalid) Enter a Non-Zero Number: ");

}

} while (size\_arr <= 0);

insert(arr, size\_arr);

printf("\nOriginal array: ");

display(arr, size\_arr);

bubbleSort(arr, size\_arr);

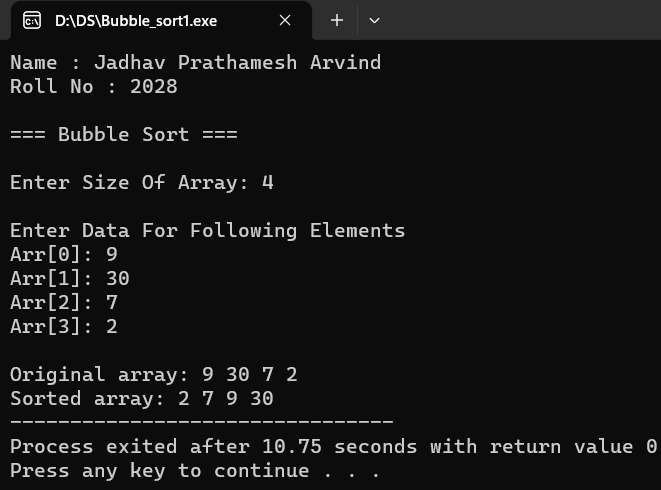
printf("\nSorted array: ");

display(arr, size\_arr);

return 0;

}

* **Output :**

****

* **Result:**

Thus, the program of bubble sort was executed successfully.

**Ex. No.: 09 9.B Implementation of Selection Sort**  **Date :**

**Aim:** To Implement Program of Selection Sort

**Algorithm:**

**1]** Algortihm selectionSort(arr, int n)

1. assign i, j, min\_idx with 0
2. loop (i < n – 1)
   1. assign min\_idx with i
   2. assign j with i + 1
   3. loop (j < n)
      1. if (arr[j] < arr[min\_idx])
         1. assign min\_idx with j
         2. swap(arr[min\_idx], arr[i])
      2. endif
      3. increment j
   4. end loop
   5. increment i
3. end loop

end selectionSort

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

void swap(int \* xp, int \* yp) {

int temp = \* xp;

\* xp = \* yp;

\* yp = temp;

}

void selectionSort(int \* arr, int n) {

int i, j, min\_idx;

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( & arr[min\_idx], & arr[i]);

}

}

void display(int \* arr, int size) {

int i;

for (i = 0; i < size; i++)

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

}

void insert(int \* arr, int size) {

int i;

printf("\nEnter Data For Following Elements\n");

for (i = 0; i < size; i++) {

printf("Arr[%d]: ", i);

scanf("%d", & arr[i]);

}

}

int main() {

int arr[50], size\_arr;

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No : 2028\n\n”);

printf("\*=\*=\*=\* Selection Sort \*=\*=\*=\*\n\n");

printf("Enter Size Of Array: ");

do {

scanf("%d", & size\_arr);

if (size\_arr <= 0) {

printf("(Invalid) Enter a Non-Zero Number: ");

}

} while (size\_arr <= 0);

insert(arr, size\_arr);

printf("\nOriginal array: ");

display(arr, size\_arr);

selectionSort(arr, size\_arr);

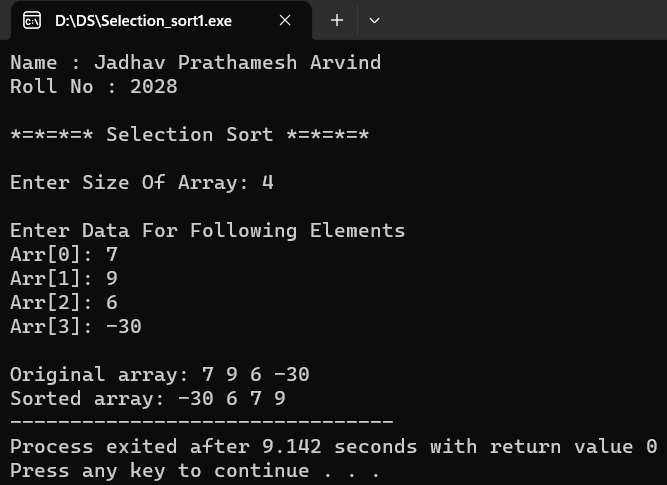
printf("\nSorted array: ");

display(arr, size\_arr);

return 0;

}

* **Output :**

****

* **Result:**

Thus, the program of selection sort was executed successfully.

**Ex. No.: 09 9.C Implementation of Insertion** **Sort**  **Date:**

**Aim:** To Implement Program of Insertion Sort

**Algorithm:**

**1]** Algorithm insertionSort (arr, n)

1. assign i with 1
2. loop (i < n)
   1. assign temp with arr[i]
   2. assign j with i - 1
   3. loop (j >= 0 && arr[j] > temp)
      1. assign arr[j + 1] with arr[j]
      2. decrement j
   4. end loop
   5. assign arr[j + 1] with temp
   6. increment i
3. end loop

end insertionSort

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

void insertionSort(int \* arr, int n) {

int i, temp, j;

for (i = 1; i < n; i++) {

temp = arr[i];

for (j = i - 1; j >= 0 && arr[j] > temp; j--)

arr[j + 1] = arr[j];

arr[j + 1] = temp;

}

}

void display(int \* arr, int n) {

int i;

for (i = 0; i < n; i++)

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

}

void insert(int \* arr, int size) {

int i;

printf("\nEnter Data For Following Elements\n");

for (i = 0; i < size; i++) {

printf("Arr[%d]: ", i);

scanf("%d", & arr[i]);

}

}

int main() {

int arr[50], size\_arr;

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No : 2028\n\n”);

printf("\*=\*=\*=\* Insertion Sort \*=\*=\*=\*\n\n");

printf("Enter Size Of Array: ");

do {

scanf("%d", & size\_arr);

if (size\_arr <= 0) {

printf("(Invalid) Enter a Non-Zero Number: ");

}

} while (size\_arr <= 0);

insert(arr, size\_arr);

printf("\nOriginal array: ");

display(arr, size\_arr);

insertionSort(arr, size\_arr);

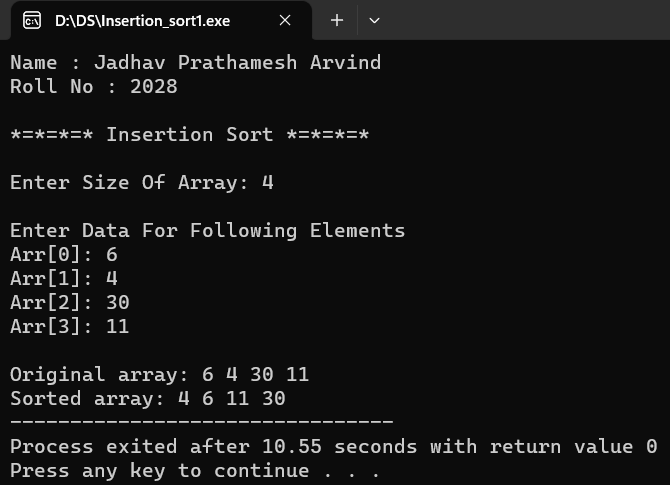
printf("\nSorted array: ");

display(arr, size\_arr);

return 0;

}

* **Output :**

****

* **Result:**

Thus, the program of insertion sort was executed successfully.

**Ex. No.: 10 10.A Implementation of Merge Sort Date:**

**Aim:** To Implement Program of Merge Sort

**Algorithm:**

**1]** Algorithm merge(arr[], left, mid, right) {

1. assign i, j and k with 0
2. assign n1 with mid - left + 1
3. assign n2 with right - mid
4. create L[n1], R[n2] array
5. loop (i < n1)
   1. assign L[i] with arr[left + i]
   2. increment i
6. end loop
7. loop (j < n2)
   1. assign R[j] with arr[mid + 1 + j]
   2. incremnet j
8. end loop
9. assign i with 0
10. assign j with 0
11. assign k with l
12. loop (i < n1 && j < n2)
    1. if (L[i] <= R[j])
       1. assign arr[k++] with L[i++]
    2. else
       1. assign arr[k++] with R[j++]
    3. end if
13. end loop
14. loop (i < n1)
    1. assign arr[k++] with L[i++]
15. end loop
16. loop (j < n2)
    1. assign arr[k++] with R[j++]
17. end loop

end merge

2] Algorithm mergeSort(arr[], left, right)

1. if (left < right)
   1. assign m with left + (right - left) / 2
   2. mergeSort(arr, left, m)
   3. mergeSort(arr, m + 1, right)
   4. merge(arr, left, m, right)
2. end if

end mergeSort

* **Program:**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

void merge(int arr[], int l, int m, int r) {

int i, j, k;

int n1 = m - l + 1;

int 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++];

else

arr[k++] = R[j++];

}

while (i < n1)

arr[k++] = L[i++];

while (j < n2)

arr[k++] = R[j++];

}

void mergeSort(int arr[], int left, int right) {

int m;

if (left < right) {

m = left + (right - left) / 2;

mergeSort(arr, left, m);

mergeSort(arr, m + 1, right);

merge(arr, left, m, right);

}

}

void insert(int array[], int sizearr) {

int i;

for (i = 0; i < sizearr; i++) {

printf("Arr[%d]: ", i);

scanf("%d", & array[i]);

}

}

void display(int \* array, int n) {

int i;

for (i = 0; i < n; i++) {

if (i == n - 1) {

printf("%d ", array[i]);

} else {

printf("%d, ", array[i]);

}

}

}

int main() {

int arr[100], i, size\_arr;

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No : 2028\n\n”);

printf("\*=\*=\*=\* Merge Sort \*=\*=\*=\*\n\n");

printf("Enter Size Of Array: ");

do {

scanf("%d", & size\_arr);

if (size\_arr <= 0) {

printf("(Invalid) Enter a Non-Zero Number: ");

}

} while (size\_arr <= 0);

printf("\nEnter Data For Following Elements: \n");

insert(arr, size\_arr);

printf("\nOriginal array: ");

display(arr, size\_arr);

mergeSort(arr, 0, size\_arr - 1);

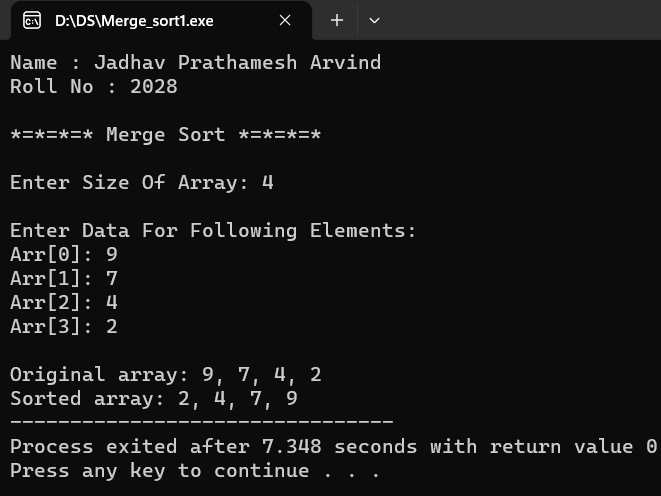
printf("\nSorted array: ");

display(arr, size\_arr);

return 0;

}

* **Output :**

****

* **Result:**

Thus, the program for merge sort was executed successsfully.

**Ex. No.: 10 10.B Implementation of Quick Sort** **Date:**

**Aim:** To Implement Program of Quick Sort

**Algorithm:**

**1]** Algorithm partition (arr[], low, high)

1. assign pivot with arr[high]
2. assign j with low
3. assign i with low - 1
4. loop (j <= high - 1)
   1. if (arr[j] <= pivot)
5. increment i
6. swap (arr[i], arr[j])
   1. end if
   2. increment j
7. end loop
8. swap (arr[i + 1], arr[high])
9. return (i + 1)

end partition

2] Algorithm quickSort (arr[], low, high)

1. if (low < high)
   1. assign pi with partition (arr, low, high)
   2. quickSort (arr, low, pi - 1)
   3. quickSort (arr, pi + 1, high)
2. end if

end quickSort

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

void swap(int \* a, int \* b) {

int t = \* a;

\* a = \* b;

\* b = t;

}

int partition(int arr[], int low, int high) {

int pivot = arr[high], j;

int i = (low - 1);

for (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 pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

void insert(int array[], int sizearr) {

int i;

for (i = 0; i < sizearr; i++) {

printf("Arr[%d]: ", i);

scanf("%d", & array[i]);

}

}

void display(int array[], int n) {

int i;

for (i = 0; i < n; i++) {

if (i == n - 1) {

printf("%d ", array[i]);

} else {

printf("%d, ", array[i]);

}

}

}

int main() {

int arr[100], size\_arr;

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No : 2028\n\n”);

printf("\*=\*=\*=\* Quick Sort \*=\*=\*=\*\n\n");

printf("Enter Size Of Array: ");

do {

scanf("%d", & size\_arr);

if (size\_arr <= 0) {

printf("(Invalid) Enter a Non-Zero Number: ");

}

} while (size\_arr <= 0);

printf("\nEnter Data For Following Elements: \n");

insert(arr, size\_arr);

printf("\nOriginal array: ");

display(arr, size\_arr);

quickSort(arr, 0, size\_arr - 1);

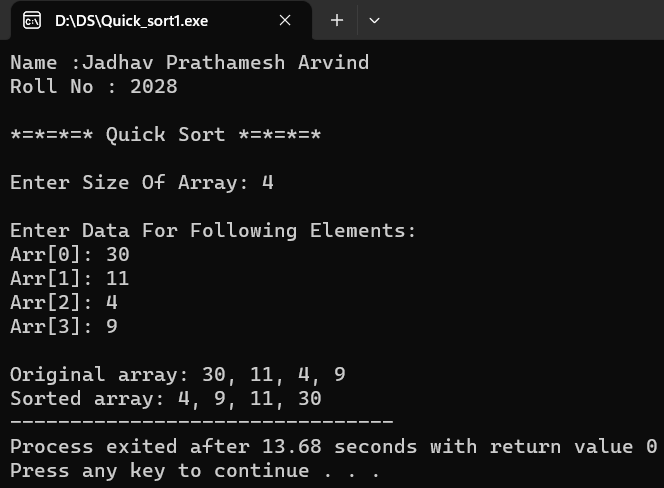
printf("\nSorted array: ");

display(arr, size\_arr);

return 0;

}

* **Output :**

****

* **Result:**

Thus**,** the program of quick sort was executed successfully.

**Ex. No.: 11 Implementation of Binary Tree Traversal Date:**

**Aim:** To Implement Program of Binary Tree Traversal

**Algorithm:**

**1]** Algorithm insert (node, data)

1. if (node equals NULL)
   1. return create\_node(data)
2. else
   1. if (data <= node -> data)
      1. assign node left with insert (node left, data)
   2. else
      1. assign node right with insert (node right, data)
   3. end if
   4. return node
3. end if

end insert

**2]** Algorithm inorderTraversal (root)

1. if (root not equals NULL)
   1. inorderTraversal (root left)
   2. display root data
   3. inorderTraversal (root right)
2. end if

end inorderTraversal

**3]** Algorithm preorderTraversa l(root)

1. if (root not equals NULL)
   1. display root data
   2. preorderTraversal (root->left);
   3. preorderTraversal (root->right);
2. end if

end preorderTraversal

**4]** Algorithm postorderTraversal (root)

1. if (root not equals NULL)
   1. postorderTraversal (root->left)
   2. postorderTraversal (root->right)
   3. display root data
2. end if

end postorderTraversal

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

struct node \*left;

struct node \*right;

};

struct node \*create\_node(int data) {

struct node \*new\_node = (struct node \*) malloc(sizeof(struct node));

new\_node->data = data;

new\_node->left = NULL;

new\_node->right = NULL;

return new\_node;

}

struct node \*insert(struct node \*node, int data) {

if (node == NULL) {

return create\_node(data);

} else {

if (data <= node -> data) {

node->left = insert(node->left, data);

} else {

node->right = insert(node->right, data);

}

return node;

}

}

void inorderTraversal(struct node \* root) {

if (root != NULL) {

inorderTraversal(root->left);

printf("%d ", root->data);

inorderTraversal(root->right);

}

}

void preorderTraversal(struct node \* root) {

if (root != NULL) {

printf("%d ", root->data);

preorderTraversal(root->left);

preorderTraversal(root->right);

}

}

void postorderTraversal(struct node \* root) {

if (root != NULL) {

postorderTraversal(root->left);

postorderTraversal(root->right);

printf("%d ", root->data);

}

}

int main() {

int n, i, data;

struct node \*root = NULL;

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No : 2028\n\n”);

printf("=\*=\*= Tree Traveral =\*=\*=\n\n");

printf("Enter Number of Nodes In The Binary Tree: ");

scanf("%d", & n);

printf("Enter Data For Following Elements: \n");

for (i = 0; i < n; i++) {

printf("Node %d: ", i + 1);

scanf("%d", &data);

root = insert(root, data);

}

printf("Inorder traversal: ");

inorderTraversal(root);\

printf("\nPreorder traversal: ");

preorderTraversal(root);

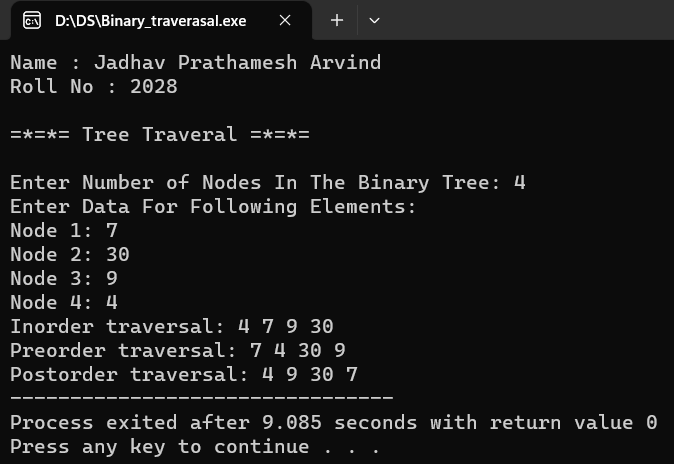
printf("\nPostorder traversal: ");

postorderTraversal(root);

return 0;

}

* **Output :**

****

* **Result:**

Thus, the program of binary tree traversal was executed successfully.

**Ex. No.: 12 Implementation of Graph Traversal Date:**

**Aim:** To Implement Program of Graph Traversal

**Algorithm:**

**1]** Algorithm add\_edge (graph, src, dest)

1. allocate (new\_node)
2. assign new\_node vertex with dest
3. assign new\_node next with graph adj\_list[src]
4. assign graph adj\_list[src] with new\_node
5. allocate (new\_node)
6. assign new\_node vertex with src
7. assign new\_node next with graph adj\_list[dest]
8. assign graph adj\_list[dest] with new\_node

end add\_edge

2] Algorithm print\_graph (graph)

1. assign i with 0
2. loop (i < graph num\_vertices)
   1. assign temp with graph adj\_list[i]
   2. loop (temp not equals NULL)
      1. display temp vertex
      2. assign temp with temp next
   3. end loop
   4. increment i
3. end loop

end print\_graph

* **Program :**

/\*Name : Jadhav Prathamesh Arvind

Roll No : 2028\*/

#include <stdio.h>

#include <stdlib.h>

#define MAX\_VERTICES 100

struct node {

int vertex;

struct node \* next;

};

struct graph {

int num\_vertices;

struct node \*\* adj\_list;

};

struct node \* create\_node(int v) {

struct node \* new\_node = (struct node \* ) malloc(sizeof(struct node));

new\_node -> vertex = v;

new\_node -> next = NULL;

return new\_node;

}

struct graph \* create\_graph(int vertices) {

struct graph \* new\_graph = (struct graph \* ) malloc(sizeof(struct graph));

new\_graph -> num\_vertices = vertices;

new\_graph -> adj\_list = (struct node \*\* ) malloc(vertices \* sizeof(struct node \* ));

int i;

for (i = 0; i < vertices; i++) {

new\_graph -> adj\_list[i] = NULL;

}

return new\_graph;

}

void add\_edge(struct graph \* g, int src, int dest) {

struct node \* new\_node = create\_node(dest);

new\_node -> next = g -> adj\_list[src];

g -> adj\_list[src] = new\_node;

new\_node = create\_node(src);

new\_node -> next = g -> adj\_list[dest];

g -> adj\_list[dest] = new\_node;

}

void print\_graph(struct graph \* g) {

int i;

printf("\n");

for (i = 0; i < g -> num\_vertices; i++) {

struct node \* temp = g -> adj\_list[i];

printf("Vertex %d: ", i);

while (temp) {

printf("%d -> ", temp -> vertex);

temp = temp -> next;

}

printf("NULL\n");

}

}

int main() {

int vertice, srcVertex, destVertex, i;

printf(“Name : Jadhav Prathamesh Arvind\n”);

printf(“Roll No : 2028\n\n”);

printf("\*=\*=\* Graph Traversal \*=\*=\*\n\nEnter Number of Vertices: ");

scanf("%d", & vertice);

struct graph \* g = create\_graph(vertice);

printf("\nEnter Data For Edges: \n");

for (i = 0; i < vertice; i++) {

printf("\nEnter Source Vertex: ");

scanf("%d", & srcVertex);

printf("Enter Destination Vertex: ");

scanf("%d", & destVertex);

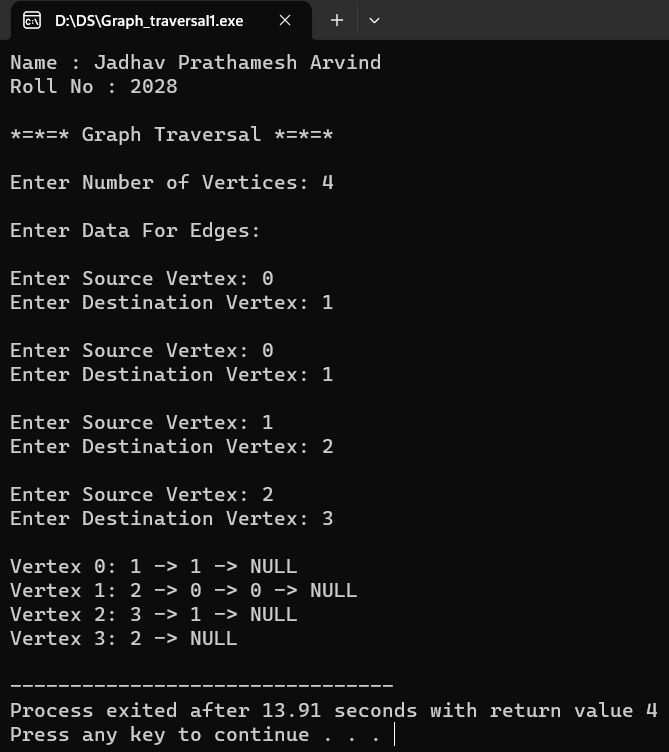
add\_edge(g, srcVertex, destVertex);

}

print\_graph(g);

}

* **Output :**

****

* **Result:**

Thus, the program for graph traversal is executed successfully.