**DATA STRUCTURE LAB FILE**

**2024-25**



**Manav Rachna International Institute of Research and Studies**

**School of Computer Applications**

**Department Of Computer Applications**

|  |  |
| --- | --- |
| **Submitted By** | |
| **Student Name** | **Noel Johnson Jacob** |
| **Roll No** | **24/SCA/BCA(AI&ML)/044** |
| **Programme** | **BCA (AI&ML)** |
| **Semester** | **2th** |
| **Section/Group** | **C** |
| **Department** | **Computer Applications** |
| **Session / Batch** | **2024-25** |
|  | |
| **Submitted To** | |
| **Faculty Name** | **Mrs Parul Gandhi / Mrs Sakshi Gupta/** |

| **Index** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **S. No** | **Program** | **CO** | **BTL** | **Date of Submission** | **Date of Evaluation** | **Remarks** | **Signature** |
|  | **DATA STRUCTURE** |  |  |  |  |  |  |
| 1. | Write a program in C to implement insertion in 1-D Arrays? |  |  |  |  |  |  |
| 2. | Write a program in C to implement deletion in 1-D Arrays ? |  |  |  |  |  |  |
| 3. | Write a program in C to concatenate two arrays? |  |  |  |  |  |  |
| 4. | Write a program in C to implement the following Operations on 2-D Array (addition; subtraction; multiplication; transpose) ? |  |  |  |  |  |  |
| 5. | Write a program in C to implement operations on Stack using array ? |  |  |  |  |  |  |
| 6. | Write a program in C to implement operations on queue using array? |  |  |  |  |  |  |
| 7. | Write a program in C to implement operations on circular queue using array? |  |  |  |  |  |  |
| 8. | Write a program in C to implement insertion in a linked list(beg; mid; end)? |  |  |  |  |  |  |
| 9. | Write a program in C to implement deletion from a linked list(beg; mid; end)? |  |  |  |  |  |  |

**Q1. Write a program in C to implement insertion in 1-D Arrays?**

**INPUT-**

#include <stdio.h>

int main() {

int arr[100], size, element, position;

printf("Enter the size of the array: ");

scanf("%d", &size);

printf("Enter the elements of the array:\n");

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

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

}

printf("Enter the element to insert: ");

scanf("%d", &element);

printf("Enter the position to insert (1 to %d): ", size + 1);

scanf("%d", &position);

for (int i = size; i >= position; i--) {

arr[i] = arr[i - 1];

}

arr[position - 1] = element; /

size++;

printf("Array after insertion:\n");

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

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

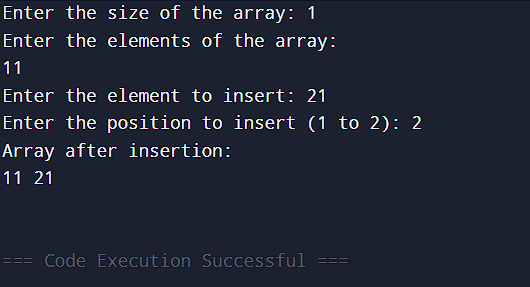
}

printf("\n");

return 0;

}

**OUTPUT-**



**Q2. Write a program in C to implement deletion in 1-D Arrays ?**

**INPUT-**

#include <stdio.h>

int main() {

int arr[100], size, position;

printf("Enter the size of the array: ");

scanf("%d", &size);

printf("Enter the elements of the array:\n");

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

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

}

printf("Enter the position to delete (1 to %d): ", size);

scanf("%d", &position);

position--;

for (int i = position; i < size - 1; i++) {

arr[i] = arr[i + 1];

}

size--;

printf("Array after deletion:\n");

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

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

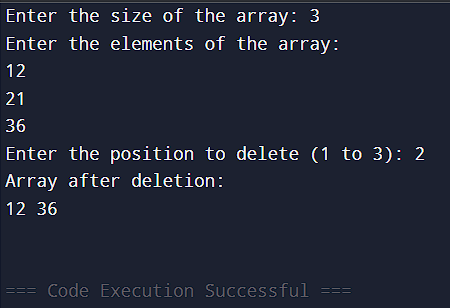
}

printf("\n");

return 0;

}

**OUTPUT-**



**Q3. Write a program in C to concatenate two arrays?**

**INPUT-**

#include <stdio.h>

void concatenateArrays(int arr1[], int size1, int arr2[], int size2, int result[]) {

for (int i = 0; i < size1; i++) {

result[i] = arr1[i];

}

for (int j = 0; j < size2; j++) {

result[size1 + j] = arr2[j];

}

}

int main() {

int size1, size2;

printf("Enter the size of the first array: ");

scanf("%d", &size1);

int arr1[size1];

printf("Enter %d elements for the first array: ", size1);

for (int i = 0; i < size1; i++) {

scanf("%d", &arr1[i]);

}

printf("Enter the size of the second array: ");

scanf("%d", &size2);

int arr2[size2];

printf("Enter %d elements for the second array: ", size2);

for (int i = 0; i < size2; i++) {

scanf("%d", &arr2[i]);

}

int result[size1 + size2];

concatenateArrays(arr1, size1, arr2, size2, result);

printf("Concatenated Array: ");

for (int k = 0; k < (size1 + size2); k++) {

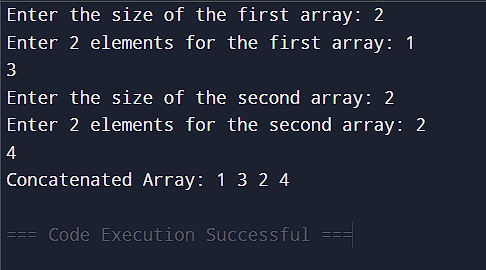
printf("%d ", result[k]);

}

return 0;

}

**OUTPUT-**



**Q4. Write a program in C to implement the following Operations on 2-D Array (addition; subtraction; multiplication; transpose) ?**

**INPUT-**

#include <stdio.h>

#define SIZE 3

void inputMatrix(int m1[SIZE][SIZE]) {

printf("Enter elements of the matrix (%d x %d):\n", SIZE, SIZE);

for (int i = 0; i < SIZE; i++) {

for (int j = 0; j < SIZE; j++) {

scanf("%d", &m1[i][j]);

}

}

}

void displayMatrix(int m2[SIZE][SIZE]) {

for (int i = 0; i < SIZE; i++) {

for (int j = 0; j < SIZE; j++) {

printf("%d ", m2[i][j]);

}

printf("\n");

}

}

void addMatrices(int a[SIZE][SIZE], int b[SIZE][SIZE], int result[SIZE][SIZE]) {

for (int i = 0; i < SIZE; i++) {

for (int j = 0; j < SIZE; j++) {

result[i][j] = a[i][j] + b[i][j];

}

}

}

void subtractMatrices(int a[SIZE][SIZE], int b[SIZE][SIZE], int result[SIZE][SIZE]) {

for (int i = 0; i < SIZE; i++) {

for (int j = 0; j < SIZE; j++) {

result[i][j] = a[i][j] - b[i][j];

}

}

}

void multiplyMatrices(int a[SIZE][SIZE], int b[SIZE][SIZE], int result[SIZE][SIZE]) {

for (int i = 0; i < SIZE; i++) {

for (int j = 0; j < SIZE; j++) {

result[i][j] = 0;

for (int k = 0; k < SIZE; k++) {

result[i][j] += a[i][k] \* b[k][j];

}

}

}

}

void transposeMatrix(int m1[SIZE][SIZE], int result[SIZE][SIZE]) {

for (int i = 0; i < SIZE; i++) {

for (int j = 0; j < SIZE; j++) {

result[j][i] = m1[i][j];

}

}

}

int main() {

int m1[SIZE][SIZE], m2[SIZE][SIZE], result[SIZE][SIZE];

printf("Matrix 1:\n");

inputMatrix(m1);

printf("Matrix 2:\n");

inputMatrix(m2);

printf("\nAddition:\n");

addMatrices(m1, m2, result);

displayMatrix(result);

printf("\nSubtraction:\n");

subtractMatrices(m1, m2, result);

displayMatrix(result);

printf("\nMultiplication:\n");

multiplyMatrices(m1, m2, result);

displayMatrix(result);

printf("\nTranspose of Matrix 1:\n");

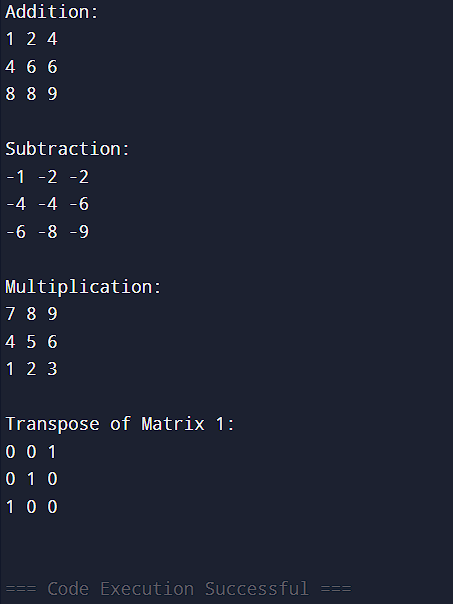
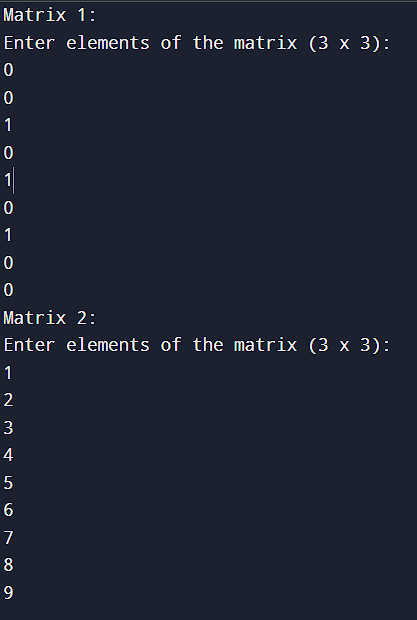
transposeMatrix(m1, result);

displayMatrix(result);

return 0;

}

**OUTPUT-**



**Q5. Write a program in C to implement operations on Stack using array ?**

**INPUT-**

#include <stdio.h>

#define MAX 100

int stack[MAX], top = -1;

void push(int value) {

if (top == MAX - 1) {

printf("Stack Overflow! Cannot insert %d\n", value);

} else {

top++;

stack[top] = value;

printf("Element %d pushed onto the stack.\n", value);

}

}

void pop() {

if (top == -1) {

printf("Stack Underflow! The stack is empty.\n");

} else {

printf("Element %d popped from the stack.\n", stack[top]);

top--;

}

}

void display() {

if (top == -1) {

printf("The stack is empty.\n");

} else {

printf("Stack elements are: ");

for (int i = top; i >= 0; i--) {

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

}

printf("\n");

}

}

int main() {

int choice, value;

while (1) {

printf("\nStack Operations:\n");

printf("1. Push\n");

printf("2. Pop\n");

printf("3. Display\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the value to push: ");

scanf("%d", &value);

push(value);

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

printf("Exiting...\n");

return 0;

default:

printf("Invalid choice! Please try again.\n");

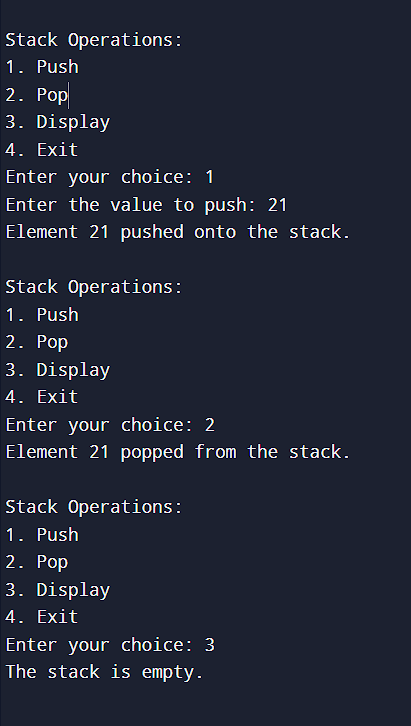
}

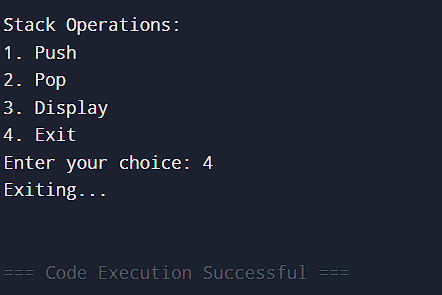
}

return 0;

}

**OUTPUT-**





**Q6. Write a program in C to implement operations on queue using array?**

**INPUT-**

#include <stdio.h>

#define SIZE 5

int queue[SIZE];

int front = -1, rear = -1;

void enqueue(int value) {

if (rear == SIZE - 1) {

printf("Queue is full!\n");

return;

}

if (front == -1) front = 0;

queue[++rear] = value;

printf("%d added to the queue\n", value);

}

void dequeue() {

if (front == -1 || front > rear) {

printf("Queue is empty!\n");

return;

}

printf("%d removed from the queue\n", queue[front++]);

if (front > rear) front = rear = -1;

}

void display() {

if (front == -1) {

printf("Queue is empty!\n");

return;

}

printf("Queue: ");

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

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

}

printf("\n");

}

int main() {

enqueue(21);

enqueue(12);

dequeue();

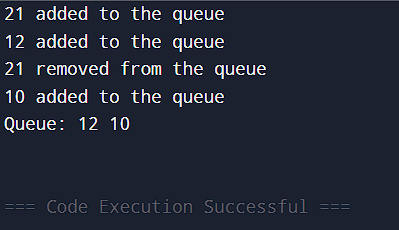
enqueue(10);

display();

return 0;

}

**OUTPUT-**



**Q7. Write a program in C to implement operations on circular queue using array?**

**INPUT-**

#include <stdio.h>

#define SIZE 5

int queue[SIZE];

int front = -1, rear = -1;

int isFull() {

return (front == 0 && rear == SIZE - 1) || (front == rear + 1);

}

int isEmpty() {

return front == -1;

}

void enqueue(int element) {

if (isFull()) {

printf("Queue is full!\n");

return;

}

if (isEmpty()) {

front = rear = 0;

} else {

rear = (rear + 1) % SIZE;

}

queue[rear] = element;

printf("%d enqueued to queue\n", element);

}

void dequeue() {

if (isEmpty()) {

printf("Queue is empty!\n");

return;

}

printf("%d dequeued from queue\n", queue[front]);

if (front == rear) {

front = rear = -1;

} else {

front = (front + 1) % SIZE;

}

}

void display() {

if (isEmpty()) {

printf("Queue is empty!\n");

return;

}

printf("Queue elements: ");

int i = front;

do {

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

i = (i + 1) % SIZE;

} while (i != (rear + 1) % SIZE);

printf("\n");

}

int main() {

enqueue(21);

enqueue(12);

enqueue(6);

enqueue(5);

enqueue(4);

display();

dequeue();

dequeue();

display();

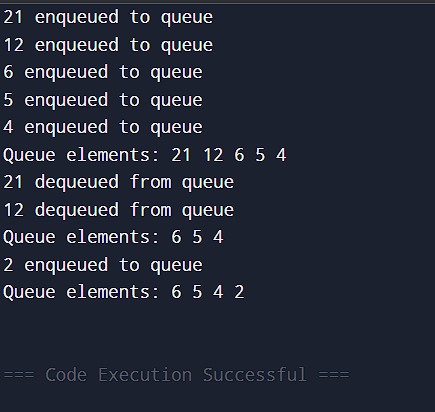
enqueue(2);

display();

return 0;

}

**OUTPUT-**



**Q8. Write a program in C to implement insertion in a linked list(beg; mid; end)?**

**INPUT-**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

void insertAtBeginning(struct Node\*\* head, int data) {

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

newNode->data = data;

newNode->next = \*head;

\*head = newNode;

}

void insertAtEnd(struct Node\*\* head, 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\* temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

void insertAtMiddle(struct Node\*\* head, int data, int position) {

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

newNode->data = data;

if (position == 1) {

insertAtBeginning(head, data);

return;

}

struct Node\* temp = \*head;

for (int i = 1; i < position - 1 && temp != NULL; i++) {

temp = temp->next;

}

if (temp == NULL) {

printf("Position out of bounds!\n");

return;

}

newNode->next = temp->next;

temp->next = newNode;

}

void displayList(struct Node\* head) {

struct Node\* temp = head;

while (temp != NULL) {

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

temp = temp->next;

}

printf("NULL\n");

}

int main() {

struct Node\* head = NULL;

insertAtBeginning(&head, 10);

insertAtEnd(&head, 20);

insertAtMiddle(&head, 15, 2);

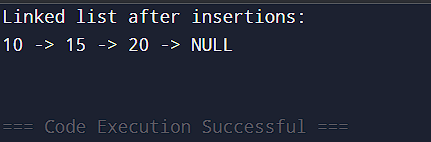
printf("Linked list after insertions:\n");

displayList(head);

return 0;

}

**OUTPUT-**



**Q9. Write a program in C to implement deletion from a linked list(beg; mid; end)?**

**INPUT-**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

void deleteBeginning(struct Node\*\* head) {

if (\*head == NULL) {

printf("List is empty!\n");

return;

}

struct Node\* temp = \*head;

\*head = (\*head)->next;

free(temp);

printf("Deleted from beginning.\n");

}

void deleteEnd(struct Node\*\* head) {

if (\*head == NULL) {

printf("List is empty!\n");

return;

}

struct Node\* temp = \*head;

struct Node\* prev = NULL;

while (temp->next != NULL) {

prev = temp;

temp = temp->next;

}

if (prev != NULL) {

prev->next = NULL;

} else {

\*head = NULL;

}

free(temp);

printf("Deleted from end.\n");

}

void deletePosition(struct Node\*\* head, int position) {

if (\*head == NULL) {

printf("List is empty!\n");

return;

}

struct Node\* temp = \*head;

if (position == 1) {

\*head = (\*head)->next;

free(temp);

printf("Deleted from position %d.\n", position);

return;

}

struct Node\* prev = NULL;

for (int i = 1; i < position && temp != NULL; i++) {

prev = temp;

temp = temp->next;

}

if (temp == NULL) {

printf("Invalid position!\n");

return;

}

prev->next = temp->next;

free(temp);

printf("Deleted from position %d.\n", position);

}

void printList(struct Node\* head) {

if (head == NULL) {

printf("List is empty!\n");

return;

}

struct Node\* temp = head;

while (temp != NULL) {

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

temp = temp->next;

}

printf("NULL\n");

}

void appendNode(struct Node\*\* head, 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\* temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

int main() {

struct Node\* head = NULL;

appendNode(&head, 10);

appendNode(&head, 20);

appendNode(&head, 30);

appendNode(&head, 40);

appendNode(&head, 50);

printf("Original List:\n");

printList(head);

deleteBeginning(&head);

printList(head);

deleteEnd(&head);

printList(head);

deletePosition(&head, 2);

printList(head);

return 0;

}

**OUTPUT-**

