***DATA STRUCTURES LAB***

***School of Computer Applications***

***Department of Computer Applications***

| ***Submitted By*** | |
| --- | --- |
| ***Student Name*** | Hansika |
| ***Roll No*** | ***24/SCA/BCA(AI&ML)/022*** |
| ***Programme*** | ***BCA (AI&ML)*** |
| ***Semester*** | ***2nd Semester*** |
| ***Section/Group*** | II C |
| ***Department*** | ***Computer Applications*** |
| ***Batch*** | ***2024-2028*** |
|  | |
| ***Submitted To*** | |
| ***Faculty Name*** | DR. Parul Gandhi |

Question-1

Write a Program in c to implement insertion in 1-D array

Input:

#include <stdio.h>

void insertElement(int arr[], int \*size, int capacity, int element, int position) {

if (\*size >= capacity) {

printf("Array is full! Cannot insert.\n");

return;

}

if (position < 1 || position > \*size + 1) {

printf("Invalid position! Position should be between 1 and %d\n", \*size + 1);

return;

}

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

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

}

arr[position - 1] = element;

(\*size)++;

}

void printArray(int arr[], int size) {

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

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

}

printf("\n");

}

int main() {

int capacity = 10;

int arr[10] = {10, 20, 30, 40, 50};

int size = 5;

printf("Original Array: ");

printArray(arr, size);

int element = 25, position = 3;

printf("Inserting %d at position %d\n", element, position);

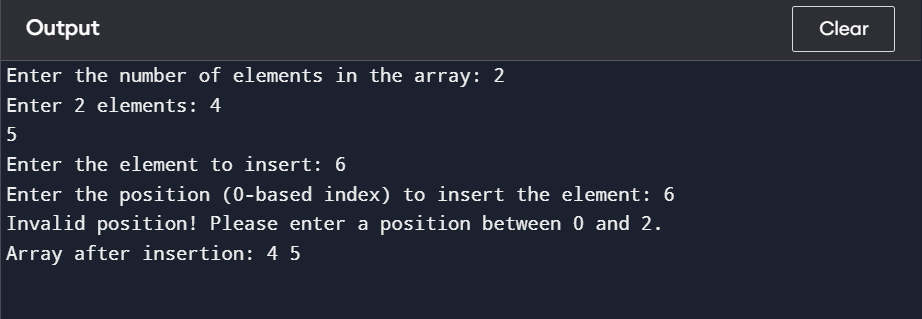
insertElement(arr, &size, capacity, element, position);

printf("Updated Array: ");

printArray(arr, size);

return 0

}



Question-2

Write a program in C to implement deletion in 1-D Arrays

Input:

include <stdio.h>

void deleteElement(int arr[], int \*size, int position) {

if (\*size == 0) {

printf("Array is empty! Cannot delete.\n");

return;

}

if (position < 1 || position > \*size) {

printf("Invalid position! Position should be between 1 and %d\n", \*size);

return;

}

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

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

}

(\*size)--;

}

void printArray(int arr[], int size) {

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

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

}

printf("\n");

}

int main() {

int arr[10] = {10, 20, 30, 40, 50}; // Initial array

int size = 5; // Current size of the array

printf("Original Array: ");

printArray(arr, size);

int position = 3;

printf("Deleting element at position %d\n", position);

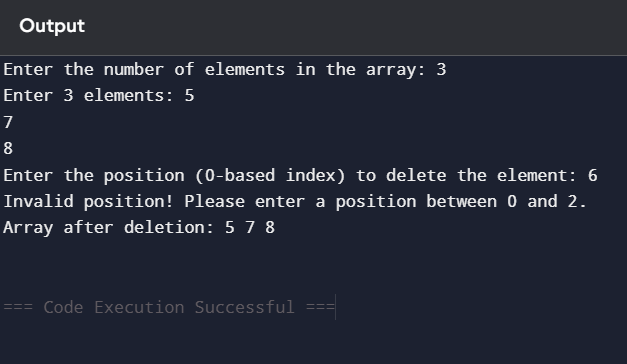
deleteElement(arr, &size, position);

printf("Updated Array: ");

printArray(arr, size);

return 0;

}



Question-3

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 i = 0; i < size2; i++) {

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

}

}

void printArray(int arr[], int size) {

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

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

}

printf("\n");

}

int main() {

int arr1[] = {1, 2, 3, 4, 5};

int size1 = sizeof(arr1) / sizeof(arr1[0]);

int arr2[] = {6, 7, 8, 9, 10};

int size2 = sizeof(arr2) / sizeof(arr2[0]);

int result[size1 + size2];

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

printf("Array 1: ");

printArray(arr1, size1);

printf("Array 2: ");

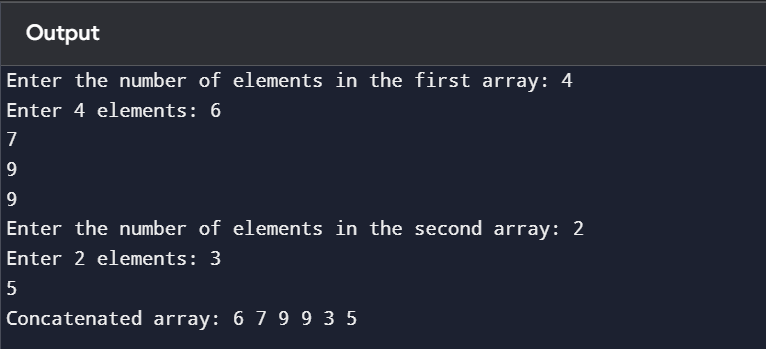
printArray(arr2, size2);

printf("Concatenated Array: ");

printArray(result, size1 + size2);

return 0;

}



Question-4

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

Input:

#include <stdio.h>

#define ROW 3

#define COL 3

void addMatrices(int A[ROW][COL], int B[ROW][COL], int result[ROW][COL]) {

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

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

result[i][j] = A[i][j] + B[i][j];

}

}

}

void subtractMatrices(int A[ROW][COL], int B[ROW][COL], int result[ROW][COL]) {

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

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

result[i][j] = A[i][j] - B[i][j];

}

}

}

void multiplyMatrices(int A[ROW][COL], int B[ROW][COL], int result[ROW][COL]) {

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

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

result[i][j] = 0;

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

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

}

}

}

}

void transposeMatrix(int A[ROW][COL], int result[ROW][COL]) {

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

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

result[j][i] = A[i][j];

}

}

}

void printMatrix(int matrix[ROW][COL]) {

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

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

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

}

printf("\n");

}

}

int main() {

int A[ROW][COL] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

int B[ROW][COL] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};

int result[ROW][COL];

printf("Matrix A:\n");

printMatrix(A);

printf("Matrix B:\n");

printMatrix(B);

addMatrices(A, B, result);

printf("Addition of matrices:\n");

printMatrix(result);

subtractMatrices(A, B, result);

printf("Subtraction of matrices:\n");

printMatrix(result);

multiplyMatrices(A, B, result);

printf("Multiplication of matrices:\n");

printMatrix(result);

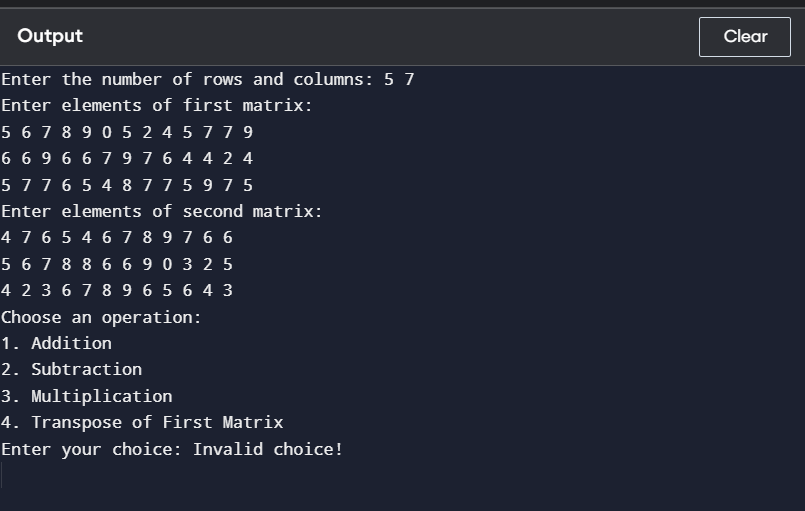
transposeMatrix(A, result);

printf("Transpose of Matrix A:\n");

printMatrix(result);

return 0;

}



Question – 5

Write a program in C to implement operations on stack using array.

Input:

#include <stdio.h>

#define MAX 5

struct Stack {

int arr[MAX];

int top;

};

void initStack(struct Stack \*s) {

s->top = -1;

}

int isEmpty(struct Stack \*s) {

return s->top == -1;

}

int isFull(struct Stack \*s) {

return s->top == MAX - 1;

}

void push(struct Stack \*s, int value) {

if (isFull(s)) {

printf("Stack is full! Cannot push %d\n", value);

return;

}

s->arr[++(s->top)] = value;

printf("Pushed %d onto stack\n", value);

}

int pop(struct Stack \*s) {

if (isEmpty(s)) {

printf("Stack is empty! Cannot pop\n");

return -1;

}

return s->arr[(s->top)--];

}

int peek(struct Stack \*s) {

if (isEmpty(s)) {

printf("Stack is empty! No top element\n");

return -1;

}

return s->arr[s->top];

}

void display(struct Stack \*s) {

if (isEmpty(s)) {

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

return;

}

printf("Stack elements: ");

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

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

}

printf("\n");

}

int main() {

struct Stack s;

initStack(&s);

push(&s, 10);

push(&s, 20);

push(&s, 30);

display(&s);

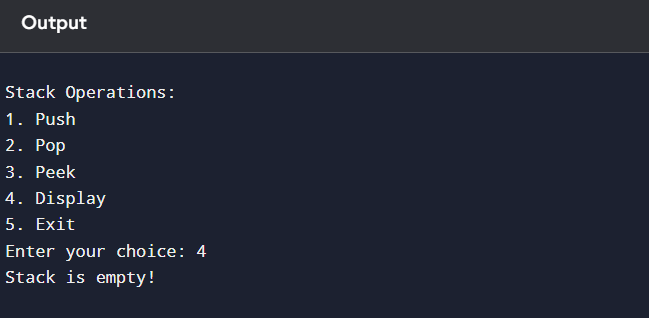
printf("Popped element: %d\n", pop(&s));

display(&s);

printf("Top element: %d\n", peek(&s));

return 0;

}



Question-6

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

Input:

#include <stdio.h>

#define MAX 5

struct Queue {

int arr[MAX];

int front, rear;

};

void initQueue(struct Queue \*q) {

q->front = -1;

q->rear = -1;

}

int isEmpty(struct Queue \*q) {

return q->front == -1;

}

int isFull(struct Queue \*q) {

return q->rear == MAX - 1;

}

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

if (isFull(q)) {

printf("Queue is full! Cannot enqueue %d\n", value);

return;

}

if (isEmpty(q)) {

q->front = 0;

}

q->arr[++(q->rear)] = value;

printf("Enqueued %d into queue\n", value);

}

int dequeue(struct Queue \*q) {

if (isEmpty(q)) {

printf("Queue is empty! Cannot dequeue\n");

return -1;

}

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

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

q->front = q->rear = -1;

} else {

q->front++;

}

return value;

}

int peek(struct Queue \*q) {

if (isEmpty(q)) {

printf("Queue is empty! No front element\n");

return -1;

}

return q->arr[q->front];

}

void display(struct Queue \*q) {

if (isEmpty(q)) {

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

return;

}

printf("Queue elements: ");

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

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

}

printf("\n");

}

int main() {

struct Queue q;

initQueue(&q);

enqueue(&q, 10);

enqueue(&q, 20);

enqueue(&q, 30);

display(&q);

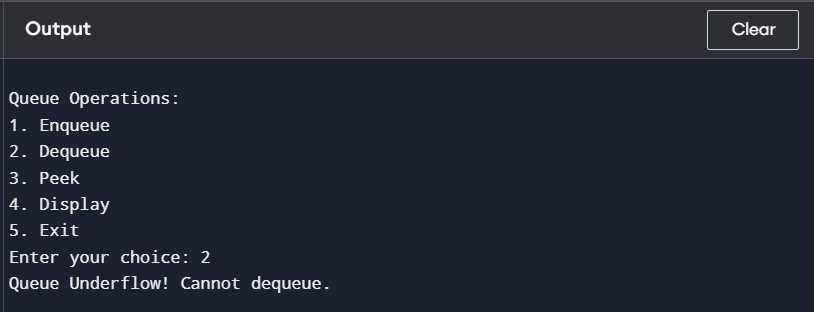
printf("Dequeued element: %d\n", dequeue(&q));

display(&q);

printf("Front element: %d\n", peek(&q));

return 0;

}



Question-7

Write a program in C to implement operations on circualar queue using array?

Input:

#include <stdio.h>

#define MAX 5

struct CircularQueue {

int arr[MAX];

int front, rear;

};

void initQueue(struct CircularQueue \*q) {

q->front = -1;

q->rear = -1;

}

int isEmpty(struct CircularQueue \*q) {

return q->front == -1;

}

int isFull(struct CircularQueue \*q) {

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

}

void enqueue(struct CircularQueue \*q, int value) {

if (isFull(q)) {

printf("Queue is full! Cannot enqueue %d\n", value);

return;

}

if (isEmpty(q)) {

q->front = q->rear = 0;

} else {

q->rear = (q->rear + 1) % MAX;

}

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

printf("Enqueued %d into queue\n", value);

}

int dequeue(struct CircularQueue \*q) {

if (isEmpty(q)) {

printf("Queue is empty! Cannot dequeue\n");

return -1;

}

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

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

q->front = q->rear = -1;

} else {

q->front = (q->front + 1) % MAX;

}

return value;

}

int peek(struct CircularQueue \*q) {

if (isEmpty(q)) {

printf("Queue is empty! No front element\n");

return -1;

}

return q->arr[q->front];

}

void display(struct CircularQueue \*q) {

if (isEmpty(q)) {

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

return;

}

printf("Queue elements: ");

int i = q->front;

while (1) {

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

if (i == q->rear) break;

i = (i + 1) % MAX;

}

printf("\n");

}

int main() {

struct CircularQueue q;

initQueue(&q);

enqueue(&q, 10);

enqueue(&q, 20);

enqueue(&q, 30);

enqueue(&q, 40);

enqueue(&q, 50);

display(&q);

printf("Dequeued element: %d\n", dequeue(&q));

display(&q);

enqueue(&q, 60);

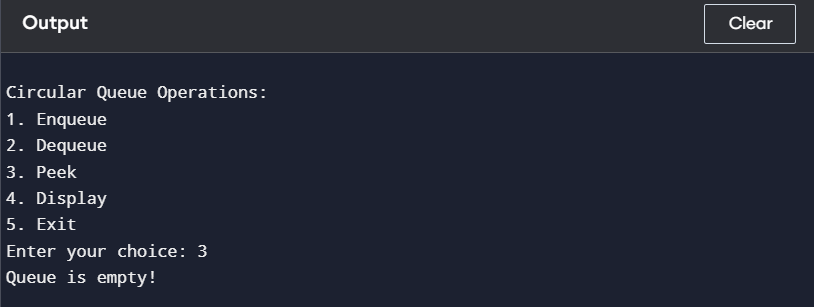
display(&q);

printf("Front element: %d\n", peek(&q));

return 0;

}

Output:



Question – 8

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 value) {

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

newNode->data = value;

newNode->next = \*head;

\*head = newNode;

}

void insertAtEnd(struct Node\*\* head, int value) {

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

newNode->data = value;

newNode->next = NULL;

if (\*head == NULL) {

\*head = newNode;

return;

}

struct Node\* temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

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

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

newNode->data = value;

if (position == 1) {

newNode->next = \*head;

\*head = newNode;

return;

}

struct Node\* temp = \*head;

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

temp = temp->next;

}

if (temp == NULL) {

printf("Position out of range\n");

free(newNode);

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);

insertAtEnd(&head, 30);

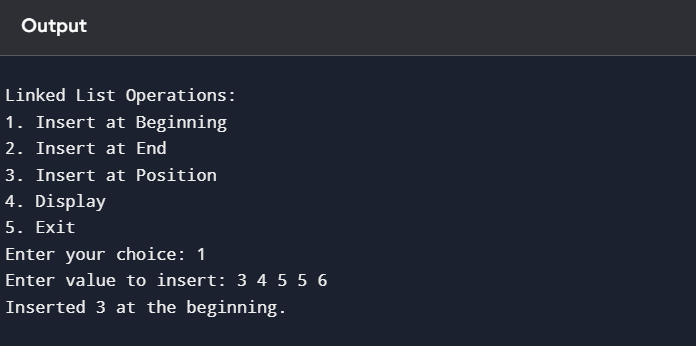
insertAtPosition(&head, 15, 2);

printf("Linked List: ");

displayList(head);

return 0;

}



Question-9

Write a program in C implement deletion from 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 value) {

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

newNode->data = value;

newNode->next = \*head;

\*head = newNode;

}

void insertAtEnd(struct Node\*\* head, int value) {

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

newNode->data = value;

newNode->next = NULL;

if (\*head == NULL) {

\*head = newNode;

return;

}

struct Node\* temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

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

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

newNode->data = value;

if (position == 1) {

newNode->next = \*head;

\*head = newNode;

return;

}

struct Node\* temp = \*head;

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

temp = temp->next;

}

if (temp == NULL) {

printf("Position out of range\n");

free(newNode);

return;

}

newNode->next = temp->next;

temp->next = newNode;

}

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

if (\*head == NULL) {

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

return;

}

struct Node\* temp = \*head;

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

free(temp);

}

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

if (\*head == NULL) {

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

return;

}

if ((\*head)->next == NULL) {

free(\*head);

\*head = NULL;

return;

}

struct Node\* temp = \*head;

while (temp->next->next != NULL) {

temp = temp->next;

}

free(temp->next);

temp->next = NULL;

}

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

if (\*head == NULL) {

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

return;

}

struct Node\* temp = \*head;

if (position == 1) {

\*head = temp->next;

free(temp);

return;

}

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

temp = temp->next;

}

if (temp == NULL || temp->next == NULL) {

printf("Position out of range\n");

return;

}

struct Node\* nextNode = temp->next->next;

free(temp->next);

temp->next = nextNode;

}

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);

insertAtEnd(&head, 30);

insertAtPosition(&head, 15, 2);

printf("Linked List after insertions: ");

displayList(head);

deleteFromBeginning(&head);

printf("Linked List after deleting from beginning: ");

displayList(head);

deleteFromEnd(&head);

printf("Linked List after deleting from end: ");

displayList(head);

deleteFromPosition(&head, 2);

printf("Linked List after deleting from position 2: ");

displayList(head);

return 0;

}

Output:

