PRACTICAL – 5

PROGRAM -1

AIM- Write A C Program For Enqueue Operation

CODE-

|  |
| --- |
| // Queue opeartions EnQueue  #include <stdio.h>  #define MAXSIZE 10 // Define the maximum size of the queue  int queue[MAXSIZE];  int front = -1;  int rear = -1;  // Function to check if the queue is full  int isFull() {  return rear == MAXSIZE - 1;  }  // Function to check if the queue is empty  int isEmpty() {  return front == -1;  }  // Function to add an element to the queue  void enqueue(int data) {  if (isFull()) {  printf("Queue Overflow\n");  } else {  if (isEmpty()) {  front = 0; // If the queue is empty, set front to 0  }  rear++;  queue[rear] = data; // Add the element to the queue  printf("%d enqueued to queue\n", data);  }  }  // Function to display the queue  void display()  {  int i;  if (isEmpty()) {  printf("Queue is empty\n");  } else {  printf("Queue elements are:\n");  for ( i = front; i <= rear; i++)  printf("%d ", queue[i]);  printf("\n");  }  }  int main()  {  printf("\n HARSH D \n");  int data;  char choice;  do {  printf("Enter the element to enqueue: ");  scanf("%d", &data);  // Enqueue the element  enqueue(data);  printf("Do you want to enqueue more elements? (y/n): ");  scanf(" %c", &choice);  } while (choice == 'y' || choice == 'Y');  // Display the queue  display();  return 0;  } |

OUTPUT-

|  |
| --- |
|  |

PROGRAM -2

AIM- Write A C Program For Dequeue Operation

CODE-

|  |
| --- |
| // Queue opeartions EnQueue  #include <stdio.h>  #define MAXSIZE 10 // Define the maximum size of the queue  int queue[MAXSIZE];  int front = 0;  int rear = -1;  // Function to check if the queue is empty  int isEmpty() {  return front > rear;  }  // Function to remove an element from the queue  int dequeue() {  if (isEmpty()) {  printf("Queue Underflow\n");  return -1; // Return -1 to indicate queue underflow  } else {  int data = queue[front]; // Get the front element  front++; // Increment front index  return data; // Return the dequeued element  }  }  // Function to add an element to the queue  void enqueue(int data) {  if (rear == MAXSIZE - 1) {  printf("Queue Overflow\n");  } else {  rear++;  queue[rear] = data; // Add the element to the queue  }  }  // Function to display the queue  void display()  {  int i;  if (isEmpty()) {  printf("Queue is empty\n");  } else {  printf("Queue elements are:\n");  for ( i = front; i <= rear; i++)  printf("%d ", queue[i]);  printf("\n");  }  }  int main()  {  printf("\n HARSH D \n");  int choice, data;  do {  printf("\nChoose an option:\n");  printf("1. Enqueue element\n");  printf("2. Dequeue element\n");  printf("3. Display queue\n");  printf("4. Exit\n");  scanf("%d", &choice);  switch (choice) {  case 1:  // Enqueue operation  printf("Enter the element to enqueue: ");  scanf("%d", &data);  enqueue(data);  break;  case 2:  // Dequeue operation  if (!isEmpty()) {  int dequeuedValue = dequeue();  printf("Dequeued: %d\n", dequeuedValue);  }  break;  case 3:  // Display operation  display();  break;  case 4:  // Exit the program  printf("Exiting...\n");  break;  default:  printf("Invalid choice. Please try again.\n");  break;  }  } while (choice != 4);  return 0;  } |

OUTPUT-

|  |
| --- |
|  |

PROGRAM -3

AIM- Write A C Program For IsEmpty Operation

CODE-

|  |
| --- |
| // Queue opeartions IsEmpty  #include <stdio.h>  #define MAXSIZE 10 // Define the maximum size of the queue  int queue[MAXSIZE];  int front = 0;  int rear = -1;  // Function to check if the queue is empty  int IsEmpty() {  return (front > rear) ? 1 : 0; // If front is greater than rear, queue is empty  }  // Function to add an element to the queue  void enqueue(int data) {  if (rear == MAXSIZE - 1) {  printf("Queue Overflow\n");  } else {  if (front == -1) { // If queue is initially empty  front = 0;  }  rear++;  queue[rear] = data;  printf("Enqueued: %d\n", data);  }  }  // Function to remove an element from the queue  void dequeue() {  if (IsEmpty()) {  printf("Queue Underflow\n");  } else {  printf("Dequeued: %d\n", queue[front]);  front++;  if (front > rear) { // Reset pointers if the queue is empty  front = 0;  rear = -1;  }  }  }  // Function to display the queue  void display()  {  int i;  if (IsEmpty()) {  printf("Queue is empty\n");  } else {  printf("Queue elements are:\n");  for (i = front; i <= rear; i++) {  printf("%d ", queue[i]);  }  printf("\n");  }  }  int main()  {  printf("\n HARSH D \n");  int choice, data;  do {  printf("\nChoose an option:\n");  printf("1. Enqueue element\n");  printf("2. Dequeue element\n");  printf("3. Display queue\n");  printf("4. Exit\n");  scanf("%d", &choice);  switch (choice) {  case 1:  // Enqueue operation  printf("Enter the element to enqueue: ");  scanf("%d", &data);  enqueue(data);  break;  case 2:  // Dequeue operation  dequeue();  break;  case 3:  // Display operation  display();  break;  case 4:  // Exit the program  printf("Exiting...\n");  break;  default:  printf("Invalid choice. Please try again.\n");  break;  }  } while (choice != 4);  return 0;  } |

OUTPUT-

|  |
| --- |
|  |

PROGRAM - 4

AIM- Write A C Program For IsFull Operation

CODE-

|  |
| --- |
| // Queue opeartions IsFull  #include <stdio.h>  #define MAXSIZE 10 // Define the maximum size of the queue  int queue[MAXSIZE];  int front = 0;  int rear = -1;  // Function to check if the queue is full  int IsFull() {  return rear == MAXSIZE - 1;  }  // Function to add an element to the queue  void enqueue(int data) {  if (IsFull()) {  printf("Queue Overflow\n");  } else {  if (front == -1) { // If queue is initially empty  front = 0;  }  rear++;  queue[rear] = data;  printf("%d enqueued to queue\n", data);  }  }  // Function to display the queue  void display()  {  int i;  if (front == -1) {  printf("Queue is empty\n");  } else {  printf("Queue elements are:\n");  for ( i = front; i <= rear; i++) {  printf("%d ", queue[i]);  }  printf("\n");  }  }  int main()  {  printf("\n HARSH D \n");  int data;  // Enqueue elements to the queue dynamically  while (!IsFull()) {  printf("Enter element to enqueue (or -1 to stop): ");  scanf("%d", &data);  if (data == -1) {  break;  }  enqueue(data);  }  // Check if the queue is full  if (IsFull()) {  printf("The queue is full.\n");  } else {  printf("The queue is not full.\n");  }  // Display the queue  display();  return 0;  } |

OUTPUT-

|  |
| --- |
|  |

PROGRAM - 5

AIM- Write A C Program For PEAK Operation in Queue

CODE-

|  |
| --- |
| // Queue opeartions PEAK  #include <stdio.h>  #define MAXSIZE 10 // Define the maximum size of the queue  int queue[MAXSIZE];  int front = 0;  int rear = -1;  // Function to check if the queue is empty  int isEmpty() {  return front > rear;  }  // Function to add an element to the queue  void enqueue(int data) {  if (rear == MAXSIZE - 1) {  printf("Queue Overflow\n");  } else {  rear++;  queue[rear] = data; // Add the element to the queue  }  }  // Function to return the value at the front of the queue without removing it (peek operation)  int peek() {  if (!isEmpty()) {  return queue[front];  } else {  printf("Queue is empty\n");  return -1; // Return -1 if the queue is empty  }  }  int main()  {  printf("\n HARSH D \n");  int data;  // Enqueue elements to the queue dynamically  while (1) {  printf("Enter element to enqueue (-1 to stop): ");  scanf("%d", &data);  if (data == -1) {  break;  }  enqueue(data);  }  // Perform the PEEK operation  int frontValue = peek();  if (frontValue != -1) {  printf("The value at the front of the queue is: %d\n", frontValue);  }  return 0;  } |

OUTPUT-

|  |
| --- |
|  |