FIRST SEMESTER MCA(2020 SCHEME) PRATICAL EXAMINATION JUNE-JULY 2021

20MCA135 DATASTRUCTURES LAB

Batch - 2

Anjaly Narayanan.
Reg:- ICE20MCA-2010

Date : 30 - June - 2021

Time : 9:30 to 12:30

1. Merging of two Arrays.
2. Implement circular queue.

Ans: 1. Algorithm

Step 1 : Start

Step 2 : Declare two arrays and then declare array 3

Step 3 : then print the size of the array 1 and Enter sorted elements of array 1. and it read.

Step 4 : Print the size of the array 2 and Enter sorted Elements of array 2 and it read.

Step 5 : Then merage array 1 and array 2.

Step 6 : After meraging we get meraged array, that stored in to array 3.

Step 7 : stop.

Program 1 : Merging two arrays.

```
#include<stdio.h>
#include<conio.h>
void main()
{
in array1[50], array2[50], array3[100], m, n, i, j, k=0.
clrscr();
Printf("\n Enter the size of the array Array1:");
Scanf("%d", &m);
printf("\n Enter sorted elements of array1:\n");
for(i=0; i<m; i++).
{
scanf("%d", &array1[i]);
}
Printf("\n Enter the size of the array Array2:");
Scanf("%d", &n);
printf("\n Enter sorted elements of array2:\n")
for(i=0; i<n; i++)
{
 scanf("%d", &array2[i]);
 }
 i=0;
 j=0;
```

```
while(i<m && j<n)
{
if(array1[i]<array2[j])
{
array3[k]=array1[i];
i++;
}
else
{
array3[k]=array2[j];
j++;
}
k++;
}
if(i>m)
{
while(j<n)
{
array3[k]=array2[j];
j++;
k++;
}}
if(j>n)
{
while(i<m)
{
array3[k]=array1[i];
i++;
k++;
```

```
    }
  }
printf("\n After merging: \n");
for(i=0; i<m+n; i++)
{
printf("\n %d", array3[i]);
}
getch();
}
```

Output

Enter the size of the array Array1 : 3

Enter the sorted elements of array 1 :

2 1 1

Enter the size the array Array 2 : 3

Enter sorted elements of array 2 :

2 1 1

After merging.

2. Program 2 : Circular Queue.

Algorithm

step 1 : start

step 2 : ~~pause~~ ~~delete~~ input the element for insertion of queue

step 3 : ~~if~~ ~~else~~ then we case case statement.

step 4 : insert, delete, display, Quit

step 5 : ~~stop~~. stop

2. Program 2 : circular queue.

```
#include <stdio.h>
#include <conio.h>
#define MAX 5
int cqueue_arr[MAX];
int front = -1;
int rear = -1;
void insert (int item)
{
if ((front == 0 && rear == MAX-1) || (front == rear+1))
{
printf ("Queue overflow \n");
return;
}
if (front == -1)
{
front = 0;
rear = 0;
}
else
{
if (rear == MAX-1)
rear = 0;
else
```

```
rear = rear+1;
}
cqueue_arr[rear] = item;
}
void deletion()
{
if (front == -1)
{
printf("Queue underflow\n");
return;
}
printf("Element deleted from queue is: %d
\n", cqueue_arr[front]);
if (front == rear)
{
front = -1;
rear = -1;
}
else
{
if (front = MAX-1)
front = 0;
else
front = front+1;
}
}
```

```
void display()
{
int front_pos = front, rear_pos = rear;
if (front == -1)
{
printf("Queue is empty \n");
return;
}
printf("Queue element \n");
if(front_pos <= rear_pos)
while (front_pos <= rear_pos)
{
printf("%d", cqueue_arr[front_pos]);
front_pos++;
 }
else
{
while (front_pos <= MAX-1)
{
printf("%d", cqueue_arr[front_pos]);
front_pos++;
}
front_pos = 0;
while (front_pos <= rear_pos)
{
```

```
printf ("%d", cqueue_arr[front_pos]);
front_pos++;
}
}
printf ("\n");
}
void main()
{
int choice, item;
clrscr();
do
{
printf ("1. Insert \n");
printf ("2. Delete \n");
printf ("3. Display \n");
printf("4. Quit \n");
printf("Enter your choice : ");
scanf ("%d", &choice);
switch (choice)
{
case 1 :
printf ("Input the element for insertion in
                                queue : ");
scanf ("%d", &item);
insert (item);
break;
```

```
case 2:
 deletion();
 break;
case 3:
 display();
 break;
 case 4:
 break;
 default:
 printf("Wrong choice \n");
 }
 }
while (choice!=4);
 getch();
 }
```

Output

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice: 1

Input the element for insertion in queue: 2

1. Insert
2. Delete

3. display
4. Quit.
Enter your choice : 1
Input the element for insertion in queue : 3
1. Insert
2. Delete.
3. display
4. Quit.
Enter your choice : 3.
Queue element
2 3.
1. Insert
2. Delete
3. Display
4. Quit
Enter your choice : 2
element deleted from queue is : 2
1. Insert
2. Delete.
3. display
4. Quit.
Enter your choice : 4