DATA STRUCTURES LAB

ETCS – 255

Faculty Name : **Asst. Prof Nitesh kr. Wadhera**

Name : **Khushi**

Enrollment No. : **03114813120**

Group : **ITE 2**

Semester : **3rd**

LAB - 1

Q1. Write a program to insert into an array and display.

Solution :

```
#include <stdio.h>

int main()
{
 int array[50], position, c, n, value;


 printf("Enter number of
 elements in the array\n");
 scanf("%d", &n);

 printf("Enter %d elements\n", n); for


 (c = 0; c < n;
 c++)
 scanf("%d",
 &array[c]);

 printf("Please enter the location
where you want to insert
annewelement\n");
 scanf("%d", &position);
```

```
 printf("Please enter
 the value\n");
 scanf("%d", &value);

 for (c = n - 1; c >=
position - 1; c--)
array[c+1] = array[c];

 array[position-1] = value;


 printf("Resultant array is\n");


 for (c = 0; c
<= n; c++)
printf("%d\n",
array[c]);


 return 0;
}
```

OUTPUT :

```
input
Enter number of elements in the array
5
Enter 5 elements
2 3 5 6 7
Please enter the location where you want to insert annewelement
4
Please enter the value
9
Resultant array is
2
3
5
9
6
7


...Program finished with exit code 0
Press ENTER to exit console.
```

Q2. Write a program to search an element from an array.

Solution:

```
#include <stdio.h>


#define MAX_SIZE 100
int main()

{

   int arr[MAX_SIZE];

   int size, i, toSearch, found;


   printf("Enter size of array: ");
   scanf("%d", &size);




   printf("Enter elements in array: ");
   for(i=0; i<size; i++)

   {

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

   }


   printf("\nEnter element to search: ");
   scanf("%d", &toSearch);
     found = 0;
```

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

   {
        if(arr[i] == toSearch)
      {
      found = 1;
       break;
      }
   }
  if(found == 1)
   {
      printf("\n%d is found at position %d", toSearch, i + 1);
   }
   Else
   {
      printf("\n%d is not found in the array", toSearch);
   }

   return 0;
}
```

OUTPUT :

```
Enter size of array: 5
Enter elements in array: 1 3 4 5 9

Enter element to search: 4

4 is found at position 3

...Program finished with exit code 0
Press ENTER to exit console.
```

Q3. Write a program to multiply two arrays in C.

Solution :

```
   int main()

 {
 int m, n, p, q, c, d, k, sum = 0; int first[10][10],

 second[10][10], multiply[10][10];

 printf("Enter the number of rows and columns of first matrix\n");
scanf("%d%d", &m, &n);

 printf("Enter the elements of first matrix\n");


 for ( c = 0 ; c < m ; c++ ) for ( d = 0 ; d < n ; d++ )
 scanf("%d", &first[c][d]);


printf("Enter the number of rows and columns of secondmatrix\n");
scanf("%d%d", &p, &q);



 if ( n != p )

 printf("Matrices with entered orders can't be
 multipliedwitheach other.\n");

 else

 {

 printf("Enter the elements of second matrix\n");
```

```
for ( c = 0 ; c < p ; c++ ) for ( d = 0 ; d < q ; d++ )
scanf("%d", &second[c][d]);

          for ( c = 0 ; c < m ; c++ )

{

           for ( d = 0 ; d < q ; d++ )
{
          for ( k = 0 ; k < p ; k++ )
{
 sum = sum + first[c][k]*second[k][d];
}
       multiply[c][d] = sum; sum = 0;

  }

}

printf("Product of entered matrices:-\n");
for ( c = 0 ; c < m ; c++ )
{
for ( d = 0 ; d < q ; d++ )
printf("%d\t", multiply[c][d]);
printf("\n");

    }

 }

   return 0;

}
```

OUTPUT:

```
Enter the number of rows and columns of first matrix
3 3
Enter the elements of first matrix
1 3 1 1 1 1 1 1 1
Enter the number of rows and columns of secondmatrix
3 3
Enter the elements of second matrix
1 1 1 1 1 1 1 1 1
Product of entered matrices:-
5       5       5
3       3       3
3       3       3


...Program finished with exit code 0
Press ENTER to exit console.
```

LAB – 2

Question no. 1: A bus can have a maximum of 55 passengers . Write a program to book the desired seat number, if available, on the bus, and also show the booking status. (i.e no. of seats booked and no. of seats available there is no need to store details of the passenger).

Solution:

```
#include <stdio.h>

int seatsAvailable = 55, seatsBooked = 0;
void bookSeat(char busPassengers[], int size, int seat)
{
  if (busPassengers[seat - 1] == 'X')
  {
    printf("Seat Occupied!!\n");
  }
  else
  {
    busPassengers[seat - 1] = 'X';
    printf("Your seat have been booked!!\n");
    printf("Enjoy your journey!!\n");
  }
  seatsAvailable--;
  seatsBooked++;
}

int main()
{
  char busPassengers[55];
  for (int i = 0; i < 55; i++)
  {
    busPassengers[i] = 'O';
  }
  int book, n;
  printf("Total number of seats available : %d\n",
seatsAvailable);
  printf("Total number of seats booked : %d\n", seatsBooked);
  int j = 1, i = 1;
  printf("Seat Numbers : \n");
  while (j != 56)
  {
    if (j < 10)
    {
      printf("0%d ", j);
    }
    else
    {
```

```
      printf("%d ", j);
    }
    if (j % 4 == 2)
    {
      printf("  ");
    }
    if (j % 4 == 0)
    {
      printf("\n");
    }

    j++;
  }
  printf("\n");
  printf("Seats that are avaiable are marked by O and seats that
are occupied are marked by X\n");
  while (i <= 55)
  {
    printf("%c ", busPassengers[i - 1]);
    if (i % 4 == 2)
    {
      printf("  ");
    }
    if (i % 4 == 0)
    {
      printf("\n");
    }
    i++;
  }
  printf("\n");
  printf("Enter total no. of seats that you want to book : ");
  scanf("%d", &n);
  for (int i = 0; i < n; i++)
  {
    printf("Enter the seat number that you want to book : ");
    scanf("%d", &book);
    bookSeat(busPassengers, 55, book);
  }
  i = 1;
  while (i <= 55)
  {
    printf("%c ", busPassengers[i - 1]);
    if (i % 4 == 2)
    {
      printf("  ");
    }
    if (i % 4 == 0)
    {
      printf("\n");
    }
    i++;
  }
```

```
  printf("\n");
  printf("Total number of seats available : %d\n",
seatsAvailable);
  printf("Total number of seats occupied : %d", seatsBooked);
  return 0;
}
```

Output Clear

```
/tmp/Raxp6QvWiq.o
Total number of seats available : 55
Total number of seats booked : 0
Seat Numbers :
01 02   03 04
05 06   07 08
09 10   11 12
13 14   15 16
17 18   19 20
21 22   23 24
25 26   27 28
29 30   31 32
33 34   35 36
37 38   39 40
41 42   43 44
45 46   47 48
49 50   51 52
53 54   55
Seats that are avaiable are marked by O and seats that are occupied are marked
    by X
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
```

Output Clear

```
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O
Enter total no. of seats that you want to book : 3
Enter the seat number that you want to book : 1 2 3
Your seat have been booked!!
Enjoy your journey!!
Enter the seat number that you want to book : Your seat have been booked!!
Enjoy your journey!!
Enter the seat number that you want to book : Your seat have been booked!!
Enjoy your journey!!
X X   X O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O O
O O   O
Total number of seats available : 52
Total number of seats occupied : 3
```

Question no. 2: Create a game application which has a box containing 10 (1 to 10 random) cards every time a user calls for a number the computer searches for the card and removes it from the box and shows the remaining cards.

Solution:

```
#include <stdio.h>

int main()
{
  int size = 10, n;
  int box[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
  int cardNumber;
  printf("Enter any card number to remove it from the box : ");
  scanf("%d", &cardNumber);
  for (int i = 0; i < 11; i++)
  {
    if (box[i] == cardNumber)
    {
      for (i; i < size; i++)
      {
        box[i] = box[i + 1];
      }
      size--;
      printf("You won!! Successfully removed the card from the
box.\n");
      printf("Remaining cards in the box : ");
    }
  }
  if (size == 10)
  {
    printf("You have choosen wrong card.Try again!!\n");
    printf("Cards in the box were ");
    for (int i = 0; i < size; i++)
    {
      printf("%d ", box[i]);
    }
    printf("\n");
  }
  else
  {
    for (int i = 0; i < size; i++)
    {
      printf("%d ", box[i]);
    }
    printf("\n");
  }
  return 0;
}
```

Output

Clear

```
/tmp/Raxp6QvWiq.o
Enter any card number to remove it from the box : 3
You won!! Successfully removed the card from the box.
Remaining cards in the box : 1 2 4 5 6 7 8 9 10
```

LAB – 3

Question 1**:** Write a program to insert an element at the beginning of the linked list and display it.

Solution:

```
#include <stdio.h>
#include <stdlib.h>
int element;
struct Node
{
    int data;
    struct Node *next;
};

struct Node *insertionAtTheBeginning(struct Node *head, int element)
{
    struct Node *ptr = (struct Node *)malloc(sizeof(struct Node));
    ptr->data = element;
    ptr->next = head;
    return ptr;
}

void display(struct Node *ptr)
{
    printf("Start-->");
    while (ptr != NULL)
    {
        printf("%d-->", ptr->data);
        ptr = ptr->next;
    }
    if (ptr == NULL)
    {
        printf("!!\n");
    }
}
int main()
{
```

```c
    int n, count = 0;
    struct Node *head = NULL;
    while (count < 3)
    {
        printf("----MENU----\n");
        printf(" 1. Insert\n 2. Display\n 3. Exit::%d\n", count + 1);
        scanf("%d", &n);
        switch (n)
        {
        case 1:
            printf("Enter the element : ");
            scanf("%d", &element);
            head = insertionAtTheBeginning(head, element);
            break;
        case 2:
            display(head);
            break;
        case 3:
            printf("Bye!!");
            break;
        default:
            printf("Invalid input!!");
            break;
        }
        count++;
    }
    return 0;
}
```

OUTPUT :

```
Output
/tmp/HpTzZ446xh.o
----MENU----
 1. Insert
 2. Display
 3. Exit::1
1
Enter the element : 10
----MENU----
 1. Insert
 2. Display
 3. Exit::2
2
Start-->1-->2-->3-->10-->!!
----MENU----
 1. Insert
 2. Display
 3. Exit::3
3
Bye!!
```

<u>Question no. 2</u> : Write a program to insert an element at the end of the linked list and display it.

Solution :

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
    int data;
    struct Node *next;
};

struct Node *insertionAtTheEnd(struct Node *head, int data)
{
    struct Node *ptr = (struct Node *)malloc(sizeof(struct Node));
    struct Node *p = head;
    while (p->next != NULL)
    {
        p = p->next;
    }
    p->next = ptr;
    ptr->data = data;
    ptr->next = NULL;
    return head;
}

void display(struct Node *ptr)
{
    printf("Start-->");
    while (ptr != NULL)
    {
        printf("%d-->", ptr->data);
        ptr = ptr->next;
    }
    if (ptr == NULL)
    {
        printf("!!\n");
    }
}
int main()
```

```c
{
    int n, element, count = 0;
    struct Node *head;
    struct Node *second;
    struct Node *third;
    head = (struct Node *)malloc(sizeof(struct Node));
    second = (struct Node *)malloc(sizeof(struct Node));
    third = (struct Node *)malloc(sizeof(struct Node));
    head->data = 1;
    second->data = 2;
    third->data = 3;
    head->next = second;
    second->next = third;
    third->next = NULL;
    while (count < 3)
    {
        printf("----MENU----\n");
        printf(" 1. Insert\n 2. Display\n 3. Exit::%d\n", count + 1);
        scanf("%d", &n);
        switch (n)
        {
        case 1:
            printf("Enter the element : ");
            scanf("%d", &element);
            head = insertionAtTheEnd(head, element);
            break;
        case 2:
            display(head);
            break;
        case 3:
            printf("Bye!!");
            break;
        default:
            printf("Invalid Input!!");
            break;
        }
        count++;
    }
    return 0;
}
```

OUTPUT:

Output Clear

```
/tmp/oVElLeDzse.o
----MENU
 1. Insert
 2. Display
 3. Exit::1
1
Enter the element : 3
----MENU
 1. Insert
 2. Display
 3. Exit::2
2
Start-->3-->!!
----MENU
 1. Insert
 2. Display
 3. Exit::3
```

LAB – 4

Question 1 : Write a program to perfom stack operations

.

Solution :

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>

struct Node
{
    int val;
    struct Node *next;
};

bool isEmpty(struct Node *top)
{
    if (top == NULL)
    {
        return true;
    }
    return false;
}

bool isFull(struct Node *top)
{
    struct Node *p = (struct Node *)malloc(sizeof(struct Node));
    if (p == NULL)
    {
        return true;
    }
    else
    {
        return false;
    }
}

struct Node *insert(struct Node *top, int x)
{
```

```c
    if (isFull(top))
    {
        printf("Stack Overflow\n");
    }
    else
    {
        struct Node *n = (struct Node *)malloc(sizeof(struct Node));
        n->val = x;
        n->next = top;
        top = n;
        return top;
    }
}

int delete (struct Node **top)
{
    if (isEmpty(*top))
    {
        printf("Stack Underflow\n");
    }
    else
    {
        struct Node *n = *top;
        *top = (*top)->next;
        int x = n->val;
        free(n);
        return x;
    }
}

void display(struct Node *ptr)
{
    while (ptr != NULL)
    {
        printf("%d ", ptr->val);
        ptr = ptr->next;
    }
    printf("\n");
}

int main()
{
    int n, element, count = 0, value;
    struct Node *top = NULL;
```

```c
    while (count < 1)
    {
        printf("----MENU----\n");
        printf(" 1. Insert\n 2. Delete\n 3. Display\n 4. Exit::%d\n", count + 1);
        scanf("%d", &n);
        switch (n)
        {
        case 1:
            printf("Enter the element : ");
            scanf("%d", &element);
            top = insert(top, element);
            break;
        case 2:
            value = delete (&top);
            printf("%d is deleted from the stack\n", value);
            break;
        case 3:
            display(top);
            break;
        case 4:
            printf("Bye!!");
            exit(0);
        default:
            printf("Invalid Input!!");
            break;
        }
    }
    return 0;
}
```

OUTPUT :

```
--- Menu ---

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

Enter your choice:1

Enter element to insert:34

--- Menu ---

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

Enter your choice:1

Enter element to insert:98

--- Menu ---

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

Enter your choice:3

Elements are...
98
34
```

```
34

--- Menu ---

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

Enter your choice:2

Deleted element is 98
--- Menu ---

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

Enter your choice:3

Elements are...
34

--- Menu ---

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

Enter your choice:4
BYE!!

...Program finished with exit code 0
Press ENTER to exit console.
```

Question 2 : Write a program to perform queue operations.

Solution :

```
#include <stdio.h>
#include <stdlib.h>

struct Node
{
    int val;
    struct Node *next;
};
struct Node *front = NULL;
struct Node *rear = NULL;

void insert(int val)
{
    struct Node *n = (struct Node *)malloc(sizeof(struct Node));
    if (n == NULL)
    {
        printf("Queue is Full");
    }
    else
    {
        n->val = val;
        n->next = NULL;
        if (front == NULL)
        {
            front = rear = n;
        }
        else
        {
            rear->next = n;
            rear = n;
        }
    }
}

int delete ()
{
```

```c
    int val = -1;
    struct Node *ptr = front;
    if (front == NULL)
    {
        printf("Queue is Empty\n");
    }
    else
    {
        front = front->next;
        val = ptr->val;
        free(ptr);
    }
    return val;
}

void display(struct Node *ptr)
{
    while (ptr != NULL)
    {
        printf("%d ", ptr->val);
        ptr = ptr->next;
    }
    printf("\n");
}

int main()
{
    int n, element, count = 0, value;

    while (count < 1)
    {
        printf("----MENU----\n");
        printf(" 1. Insert\n 2. Delete\n 3. Display\n 4. Exit::%d\n", count + 1);
        scanf("%d", &n);
        switch (n)
        {
        case 1:
            printf("Enter the element : ");
            scanf("%d", &element);
            insert(element);
            break;
        case 2:
            value = delete ();
            printf("%d is deleted from the stack\n", value);
            break;
```

```
        case 3:
            display(front);
            break;
        case 4:
            printf("Bye!!");
            exit(0);
        default:
            printf("Invalid Input!!");
            break;
        }
    }
    return 0;
}
```

SOLUTION :

```
Input
Enter 1 to insert element in the queue
Enter 2 to delete element from the queue
Enter 3 to display elemets of the queue
Enter 4 to exit
1
Enter the element you want to push in the stack
56
Enter 1 to insert element in the queue
Enter 2 to delete element from the queue
Enter 3 to display elemets of the queue
Enter 4 to exit
1
Enter the element you want to push in the stack
76
Enter 1 to insert element in the queue
Enter 2 to delete element from the queue
Enter 3 to display elemets of the queue
Enter 4 to exit
3
56 76
Enter 1 to insert element in the queue
Enter 2 to delete element from the queue
Enter 3 to display elemets of the queue
Enter 4 to exit
2
Enter 1 to insert element in the queue
Enter 2 to delete element from the queue
Enter 3 to display elemets of the queue
Enter 4 to exit
3
76
Enter 1 to insert element in the queue
Enter 2 to delete element from the queue
Enter 3 to display elemets of the queue
Enter 4 to exit
4
4
BYE!!

...Program finished with exit code 0
Press ENTER to exit console.
```

LAB – 5

Question 1 : Write a program to perform stack operations using arrays.

Solution :

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

structStack
{
    int top;
    int size;
    int*arr;
};

boolis_empty(structStack*s)
{
    return(s->top <0);
}
boolis_Full(structStack*s)
{
    return(s->top >=s->size -1);
}
voidpush(structStack*s,intdata)
{
    if(is_Full(s))
    {
        printf("Stack Overflow\n");
    }
    else
    {
        s->top =s->top +1;
        s->arr[s->top]=data;

    }
}
```

```
intpop(structStack*s)
{
    if(is_empty(s))
    {
        printf("Stack Underflow\n");
        return-1;
    }
    else
    {
        intval;
        val=s->arr[s->top--];
        returnval;
    }
}

intpeek(structStack*s)
{
    if(s->top <0)
    {
        printf("Stack is empty\n");
        return-1;
    }
    else
    {
        inta=s->arr[s->top];
        returna;
    }
}
voiddisplay(structStack*s)
{
    if(is_empty(s))
    {
        printf("Stack Underflow\n");
    }
    else
    {
        while(!is_empty(s))
        {
            printf("%d",peek(s));
            pop(s);
        }
        printf("\n");
    }
}
intmain()
```

```
{
    intn,element,count=0,value;
    structStack*s;
    s->size =10;
    s->top =-1;
    s->arr=(int*)malloc(sizeof(s->size));
    while(count<1)
    {
        printf("----MENU----\n");
        printf("1. Push\n2. Pop\n3. Peek\n4. Display and pop stack data\n5.
Exit::%d\n",count+1);
        scanf("%d",&n);
        switch(n)
        {
        case1:
            printf("Enter the element : ");
            scanf("%d",&element);
            push(s,element);
            printf("%d is pushed into the stack\n",element);
            break;
        case2:
            value=pop(s);
            printf("%d is popped from the stack\n",value);
            break;
        case3:
            printf("Topmost element is %d\n",peek(s));
            break;
        case4:
            display(s);
            break;
        case5:
            printf("Bye!!");
            exit(0);
        default:
            printf("Invalid Input!!");
            break;
        }
    }
    return0;
}
```

OUTPUT:

```
----MENU----
1. Push
2. Pop
3. Peek
4. Display and pop stack data
5. Exit::1
1
Enter the element : 10
10 is pushed into the stack
----MENU----
1. Push
2. Pop
3. Peek
4. Display and pop stack data
5. Exit::1
1
Enter the element : 20
20 is pushed into the stack
----MENU----
1. Push
2. Pop
3. Peek
4. Display and pop stack data
5. Exit::1
1
Enter the element : 30
30 is pushed into the stack
----MENU----
1. Push
2. Pop
3. Peek
4. Display and pop stack data
5. Exit::1
2
30 is popped from the stack
----MENU----
1. Push
2. Pop
3. Peek
4. Display and pop stack data
5. Exit::1
3
Topmost element is 20
----MENU----
1. Push
2. Pop
3. Peek
4. Display and pop stack data
5. Exit::1
4
20 10
----MENU----
1. Push
2. Pop
3. Peek
4. Display and pop stack data
5. Exit::1
5
Bye!!
Process returned 0 (0x0)   execution time : 39.176 s
Press any key to continue.
```

Question 2 : Write a program to perform Queue operations using arrays.

Solution :

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

structcircularQueue
{
    int*arr, MAX_SIZE, front, rear;
};

boolisEmpty(structcircularQueue*q)
{
    if(q->front ==-1)
    {
        returntrue;
    }
    returnfalse;
}
boolisFull(structcircularQueue*q)
{
    if((q->rear +1)%q->MAX_SIZE ==q->front)
    {
        returntrue;
    }
    returnfalse;
}
voidenQueue(structcircularQueue*q,intval)
{
    if(isFull(q))
    {
        printf("Queue is full.\n");
    }
    else
    {
        if(q->front ==-1)
        {
            q->front =0;
        }
        q->rear =(q->rear +1)%q->MAX_SIZE;
```

```
        q->arr[q->rear]=val;
    }
}
intdeQueue(structcircularQueue*q)
{
    intval=0;
    if(isEmpty(q))
    {
        printf("Queue is empty.\n");
        return-1;
    }
    else
    {
        val=q->arr[q->front];
        if(q->front ==q->rear)
        {
            q->front =-1;
            q->rear =-1;
        }
        else
        {
            q->front =(q->front +1)%q->MAX_SIZE;
        }
    }
    returnval;
}
voiddisplay(structcircularQueue*q)
{
    if(isEmpty(q))
    {
        printf("Queue is empty\n");
    }
    else
    {
        inti=q->front;
        for(i;i!=q->rear;i=(i+1)%q->MAX_SIZE)
        {
            printf("%d",q->arr[i]);
        }
        printf("%d\n",q->arr[i]);
    }
}
intmain()
{
    intn,element,count=0,value;
```

```
    structcircularQueue*q;
    q->MAX_SIZE =10;
    q->arr=(int*)malloc(q->MAX_SIZE *sizeof(int));
    q->front =-1;
    q->rear =-1;
    while(count<1)
    {
        printf("----MENU----\n");
        printf(" 1. Enqueue\n 2. Dequeue\n 3. Display\n 4. Exit::%d\n",count+1);
        scanf("%d",&n);
        switch(n)
        {
        case1:
            printf("Enter the element : ");
            scanf("%d",&element);
            enQueue(q,element);
            break;
        case2:
            value=deQueue(q);
            printf("%d is dequeued from the Queue\n",value);
            break;
        case3:
            display(q);
            break;
        case4:
            printf("Bye!!");
            exit(0);
        default:
            printf("Invalid Input!!");
            break;
        }
    }
    return0;
}
```

OUTPUT:

```
----MENU----
 1. Enqueue
 2. Dequeue
 3. Display
 4. Exit::1
1
Enter the element : 30
----MENU----
 1. Enqueue
 2. Dequeue
 3. Display
 4. Exit::1
1
Enter the element : 20
----MENU----
 1. Enqueue
 2. Dequeue
 3. Display
 4. Exit::1
1
Enter the element : 10
----MENU----
 1. Enqueue
 2. Dequeue
 3. Display
 4. Exit::1
3
30 20 10
----MENU----
 1. Enqueue
 2. Dequeue
 3. Display
 4. Exit::1
2
30 is dequeued from the Queue
----MENU----
 1. Enqueue
 2. Dequeue
 3. Display
 4. Exit::1
3
20 10
----MENU----
 1. Enqueue
 2. Dequeue
 3. Display
 4. Exit::1
4
Bye!!
```

Question No.11-Write a program to perform queue operations using arrays (Using menu-driven approach).

Sol. INPUT:

```
#include <stdio.h>
#include<stdlib.h>
#define MAX 50
int insert();
int delete();
int display();
int queue_array[MAX];
int rear = - 1;
int front = - 1;
int main()
{
int choice;
while (1)
{
printf("1.Insert element to queue \n");
printf("2.Delete element from queue \n");
printf("3.Display all elements of queue \n");
printf("4.Quit \n");
printf("Enter your choice : ");
scanf("%d", &choice);
switch (choice)
{
case 1:
insert();
break;
case 2:
delete();
break;
case 3:
display();
break;
case 4:
exit(1);
default:
printf("Wrong choice \n");
}
}
```

```
}
int insert()
{
int add_item;
if (rear == MAX - 1)
printf("Queue Overflow \n");
else
{
if (front == - 1)
front = 0;
printf("Inset the element in queue : ");
scanf("%d", &add_item);
rear = rear + 1;
queue_array[rear] = add_item;
}
}
int delete()
{
if (front == - 1 || front > rear)
{
printf("Queue Underflow \n");
return 0 ;
}
else
{
printf("Element deleted from queue is : %d\n", queue_array[front]);
front = front + 1;
}
}
int display()
{
int i;
if (front == - 1)
printf("Queue is empty \n");
else
{
printf("Queue is : \n");
for (i = front; i <= rear; i++)
printf("%d ", queue_array[i]);
printf("\n");
}
return 0;}
```

OUTPUT:

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 25
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 34
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 2
Element deleted from queue is : 25
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :
34
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 4
```

LAB-6

Question No.12- Create a doubly-linked list with nodes having information about an employee and perform Insertion in front of doubly linked list and perform deletion at the end of that doubly linked list. (Use a menu-driven approach).

Solution: INPUT:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Node
{
    char name[20];
    int age;
    long employeeId;
    float salary;
    unsigned long long phoneNo;
    struct Node *next;
    struct Node *prev;
};

struct Node *insertion(struct Node *head, char name[20], int age,
long int employeeId, unsigned long long PhoneNo, float salary)
{
    struct Node *ptr = (struct Node *)malloc(sizeof(struct Node));
    printf("Enter Name of employee : ");
    scanf("%s", name);
    printf("Enter age of employee : ");
    scanf("%d", &age);
    printf("Enter employeeId of employee : ");
    scanf("%ld", &employeeId);
    printf("Enter salary of employee : ");
    scanf("%f", &salary);
    printf("Enter phoneNo. of employee : ");
    scanf("%llu", &PhoneNo);
    strcpy(ptr->name, name);
    ptr->age = age;
    ptr->employeeId = employeeId;
    ptr->phoneNo = PhoneNo;
```

```
    ptr->salary = salary;
    ptr->next = head;
    ptr->prev = NULL;
    head->prev = ptr;
    head = ptr;
    return head;
}

void deletion(struct Node *head)
{
    struct Node *p = head;
    struct Node *ptr = head->next;
    while (ptr->next != NULL)
    {
        ptr = ptr->next;
        p = p->next;
    }
    long val = p->employeeId;
    printf("Employee with employee id %ld's data is deleted from the
list.\n", val);
    p->next = NULL;
    free(ptr);
}

void display(struct Node *head)
{
    int count = 1;
    while (head->next != NULL)
    {
        printf("Displaying employee no. %d's data : \n\n", count);
        printf("Name : %s\n", head->name);
        printf("Age : %d\n", head->age);
        printf("Employee id : %ld\n", head->employeeId);
        printf("Phone No. : %llu\n", head->phoneNo);
        printf("salary : %0.2f\n\n", head->salary);
        head = head->next;
        count++;
    }
}
int main()
{
    int n, count = 0, age;
```

```
    long int employeId;
    float salary;
    char name[20];
    unsigned long long PhoneNo;
    struct Node *head = (struct Node *)malloc(sizeof(struct Node));
    head->next = NULL;
    head->prev = NULL;
    while (count < 1)
    {
        printf("----MENU----\n");
        printf("1. Insert\n2. Delete\n3. Display\n4. Exit::%d\n",
count + 1);
        scanf("%d", &n);
        switch (n)
        {
        case 1:
            head = insertion(head, name, age, employeId, PhoneNo,
salary);
            break;
        case 2:
            deletion(head);
            break;
        case 3:
            display(head);
            break;
        case 4:
            printf("Bye!!");
            exit(0);
            break;
        default:
            printf("Invalid input!!");
            break;
        }
    }
    return 0;
}
```

OUTPUT:

```
PS G:\college code> cd "g:\college code\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
----MENU----
1. Insert
2. Delete
3. Display
4. Exit::1
1
Enter Name of employee : ab
Enter age of employee : 24
Enter employeeId of employee : 1254
Enter salary of employee : 2364467
Enter phoneNo. of employee : 4532637
----MENU----
1. Insert
2. Delete
3. Display
4. Exit::1
1
Enter Name of employee : cd
Enter age of employee : 60
Enter employeeId of employee : 7598
Enter salary of employee : 93579305
Enter phoneNo. of employee : 31845739
----MENU----
1. Insert
2. Delete
3. Display
4. Exit::1
3
Displaying employee no. 1's data :

Name : cd
Age : 60
Employee id : 7598
Phone No. : 31845739
salary : 93579304.00

Displaying employee no. 2's data :

Name : ab
Age : 24
Employee id : 1254
Phone No. : 4532637
salary : 2364467.00

----MENU----
```

```
3
Displaying employee no. 1's data :

Name : cd
Age : 60
Employee id : 7598
Phone No. : 31845739
salary : 93579304.00

Displaying employee no. 2's data :

Name : ab
Age : 24
Employee id : 1254
Phone No. : 4532637
salary : 2364467.00

----MENU----
1. Insert
2. Delete
3. Display
4. Exit::1
2
Employee with employee id 1254's data is deleted from the list.
----MENU----
1. Insert
2. Delete
3. Display
4. Exit::1
3
Displaying employee no. 1's data :

Name : cd
Age : 60
Employee id : 7598
Phone No. : 31845739
salary : 93579304.00

----MENU----
1. Insert
2. Delete
3. Display
4. Exit::1
4
Bye!!
PS G:\college code>
```

Question No.13- Create a circular linked list having information about a college and perform insertion in front and perform deletion at the end.

Solution- INPUT:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Node
{
    long int rank;
    char name[30];
    int totalStudents, totalFaculties, totalLabs;
    struct Node *next;
};

struct Node *insertion(struct Node *head, long int rank, char
name[30], int totalStudents, int totalFaculties, int totalLabs)
{
    struct Node *ptr = (struct Node *)malloc(sizeof(struct Node));
    struct Node *q = head->next;
    printf("Enter Name of college : ");
    scanf("%s", name);
    printf("Enter rank of college : ");
    scanf("%ld", &rank);
    printf("Enter how many students there are in the college : ");
    scanf("%d", &totalStudents);
    printf("Enter total number of faculties avaiable in the college
: ");
    scanf("%d", &totalFaculties);
    printf("Enter total number of labs available in the college :
");
    scanf("%d", &totalLabs);
    strcpy(ptr->name, name);
    ptr->rank = rank;
    strcpy(ptr->name, name);
    ptr->rank = rank;
    ptr->totalFaculties = totalFaculties;
    ptr->totalLabs = totalLabs;
    ptr->totalStudents = totalStudents;
    while (q->next != head)
```

```
    {
        q = q->next;
    }
    // At this point q points to the last node of the circular
linked list.
    q->next = ptr;
    ptr->next = head;
    head = ptr;
    return head;
}

void deletion(struct Node *head)
{
    struct Node *p = head;
    if (p->next == NULL)
    {
        printf("List is empty\n");
        return;
    }
    struct Node *ptr = head->next;
    while (ptr->next != head)
    {
        ptr = ptr->next;
        p = p->next;
    }
    char val[30];
    strcpy(val, ptr->name);
    free(ptr);
    printf("College with name %s's data is deleted from the list\n",
val);
    p->next = head;
}

void display(struct Node *head)
{
    struct Node *ptr = head;
    int count = 1;
    do
    {
        printf("Displaying details of college %d : \n\n", count);
        printf("Name : %s\n", ptr->name);
        printf("rank : %ld\n", ptr->rank);
```

```
        printf("Total students : %d\n", ptr->totalStudents);
        printf("Total Faculties : %d\n", ptr->totalFaculties);
        printf("Total labs : %d\n\n", ptr->totalLabs);
        ptr = ptr->next;
        count++;
    } while (ptr != head);
}

int main()
{
    int n, count = 0;
    long int rank;
    char name[30];
    int totalStudents, totalFaculties, totalLabs;
    struct Node *head = (struct Node *)malloc(sizeof(struct Node));
    struct Node *second = (struct Node *)malloc(sizeof(struct
Node));
    strcpy(head->name, "abcd");
    head->rank = 10;
    head->totalFaculties = 55;
    head->totalLabs = 13;
    head->totalStudents = 1200;
    head->next = second;
    strcpy(second->name, "efgh");
    second->totalFaculties = 50;
    second->totalLabs = 15;
    second->totalStudents = 1000;
    second->rank = 15;
    second->next = head;
    while (count < 1)
    {
        printf("----MENU----\n");
        printf("1. Insert\n2. Delete\n3. Display\n4. Exit::%d\n",
count + 1);
        scanf("%d", &n);
        switch (n)
        {
        case 1:
            head = insertion(head, rank, name, totalStudents,
totalFaculties, totalLabs);
            break;
        case 2:
```

```
            deletion(head);
            break;
        case 3:
            display(head);
            break;
        case 4:
            printf("Bye!!");
            exit(0);
            break;
        default:
            printf("Invalid input!!");
            break;
        }
    }
    return 0;
}
```

OUTPUT:

```
----MENU----
1. Insert
2. Delete
3. Display
4. Exit::1
1
Enter Name of college : DSNU
Enter rank of college : 4
Enter how many students there are in the college : 3566
Enter total number of faculties avaiable in the college : 78495
Enter total number of labs available in the college : 64
----MENU----
1. Insert
2. Delete
3. Display
4. Exit::1
1
Enter Name of college : AIGJU
Enter rank of college : 5
Enter how many students there are in the college : 1455
Enter total number of faculties avaiable in the college : 234
Enter total number of labs available in the college : 12
----MENU----
1. Insert
2. Delete
3. Display
4. Exit::1
3
Displaying details of college 1 :

Name : AIGJU
rank : 5
Total students : 1455
Total Faculties : 234
Total labs : 12

Displaying details of college 2 :

Name : DSNU
rank : 4
Total students : 3566
Total Faculties : 78495
Total labs : 64

Displaying details of college 3 :
```

```
Name : abcd
rank : 10
Total students : 1200
Total Faculties : 55
Total labs : 13 :

Displaying details of college 4 :

Name : efgh
rank : 15
Total students : 1000
Total Faculties : 30
Total labs : 15

----MENU----
1. Insert
2. Delete
3. Display
4. Exit::1
2
College with name efgh's data is deleted from the list
----MENU----
1. Insert
2. Delete
3. Display
4. Exit::1
3
Displaying details of college 1 :

Name : AIGJU
rank : 5
Total students : 1455
Total Faculties : 234
Total labs : 12

Displaying details of college 2 :

Name : DSNU
rank : 4
Total students : 3566
Total Faculties : 78495
Total labs : 64

Displaying details of college 3 :
```

```
----MENU----
1. Insert
2. Delete
3. Display
4. Exit::1
2
College with name efgh's data is deleted from the list
----MENU----
1. Insert
2. Delete
3. Display
4. Exit::1
3
Displaying details of college 1 :

Name : AIGJU
rank : 5
Total students : 1455
Total Faculties : 234
Total labs : 12

Displaying details of college 2 :

Name : DSNU
rank : 4
Total students : 3566
Total Faculties : 78495
Total labs : 64

Displaying details of college 3 :

Name : abcd
rank : 10
Total students : 1200
Total Faculties : 55
Total labs : 13

----MENU----
1. Insert
2. Delete
3. Display
4. Exit::1
4
Bye!!
PS G:\college code>
```

LAB – 7

Question No.14-: Create a binary search tree (display using graphics) perform tree traversals (preorder , postorder , inorder) using the concept of recursion.

Solution- INPUT:

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
    int data;
    struct Node *right;
    struct Node *left;
};

void inOrderTraversal(struct Node *p)
{
    if (p == NULL)
    {
        return;
    }
    inOrderTraversal(p->left);
    printf("%d ", p->data);
    inOrderTraversal(p->right);
}

void postOrderTraversal(struct Node *p)
{
    if (p == NULL)
    {
        return;
    }
    postOrderTraversal(p->left);
    postOrderTraversal(p->right);
    printf("%d ", p->data);
}

void preOrderTraversal(struct Node *p)
{
```

```
    if (p == NULL)
    {
        return;
    }
    printf("%d ", p->data);
    preOrderTraversal(p->left);
    preOrderTraversal(p->right);
}

int main()
{
    int n, count = 0;
    struct Node *root = (struct Node *)malloc(sizeof(struct Node));
    struct Node *second = (struct Node *)malloc(sizeof(struct
Node));
    struct Node *third = (struct Node *)malloc(sizeof(struct Node));
    struct Node *fourth = (struct Node *)malloc(sizeof(struct
Node));
    struct Node *fifth = (struct Node *)malloc(sizeof(struct Node));
    struct Node *sixth = (struct Node *)malloc(sizeof(struct Node));
    struct Node *seventh = (struct Node *)malloc(sizeof(struct
Node));
    struct Node *eighth = (struct Node *)malloc(sizeof(struct
Node));
    root->data = 4;
    root->left = second;
    root->right = third;
    second->data = 2;
    second->left = fourth;
    second->right = fifth;
    third->data = 8;
    third->left = sixth;
    third->right = seventh;
    fourth->data = 1;
    fourth->left = NULL;
    fourth->right = NULL;
    fifth->data = 3;
    fifth->left = NULL;
    fifth->right = NULL;
    sixth->data = 5;
    sixth->left = NULL;
    sixth->right = NULL;
```

```
    seventh->data = 9;
    seventh->left = eighth;
    seventh->right = NULL;
    eighth->data = 7;
    eighth->left = NULL;
    eighth->right = NULL;
    while (count < 1)
    {
        printf("----MENU----\n");
        printf("1. Preorder Traversal\n2. Postorder Traversal\n3.
InOrder Traversal\n4. Exit::%d\n", count + 1);
        scanf("%d", &n);
        switch (n)
        {
        case 1:
            preOrderTraversal(root);
            printf("\n");
            break;
        case 2:
            postOrderTraversal(root);
            printf("\n");
            break;
        case 3:
            inOrderTraversal(root);
            printf("\n");
            break;
        case 4:
            printf("Bye!!");
            exit(0);
            break;

        default:
            printf("Invalid input!!");
            break;
        }
    }
    return 0;
}
```

OUTPUT:

```
PS G:\college code> cd "g:\college code\" ; if ($?) { gcc 11_15.c -o 11_15 } ; if ($?) { .\11_15 }
----MENU----
1. Preorder Traversal
2. Postorder Traversal
3. InOrder Traversal
4. Exit::1
1
4 2 1 3 8 5 9 7
----MENU----
1. Preorder Traversal
2. Postorder Traversal
3. InOrder Traversal
4. Exit::1
2
1 3 2 5 7 9 8 4
----MENU----
1. Preorder Traversal
2. Postorder Traversal
3. InOrder Traversal
4. Exit::1
3
1 2 3 4 5 8 7 9
----MENU----
1. Preorder Traversal
2. Postorder Traversal
3. InOrder Traversal
4. Exit::1
4
Bye!!
PS G:\college code>
```

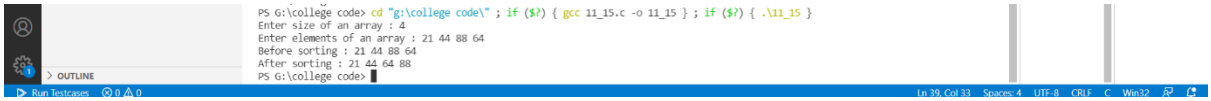
Question No.15-Write a program to implement the insertion sort using array as a data structure.

Solution- INPUT:

```
#include <stdio.h>

void insertionSort(int arr[], int size)
{
    int j, key;
    for (int i = 1; i < size; i++)
    {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key)
        {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

void printArray(int arr[], int size)
{
    for (int i = 0; i < size; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main()
{
    int n;
    printf("Enter size of an array : ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter elements of an array : ");
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
```

```
    }
    printf("Before sorting : ");
    printArray(arr, n);
    printf("After sorting : ");
    insertionSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

OUTPUT:


PS G:\college code> cd "g:\college code\" ; if ($?) { gcc 11_15.c -o 11_15 } ; if ($?) { .\11_15 }
Enter size of an array : 4
Enter elements of an array : 21 44 88 64
Before sorting : 21 44 88 64
After sorting : 21 44 64 88
PS G:\college code>
OUTLINE
Run Testcases
Ln 39, Col 33
Spaces: 4
UTF-8
CRLF
C
Win32

LAB-8

Question 1: Write a program to implement selection sort using array as a data structure.

Source Code:

```
#include <stdio.h>
void printArray(int *arr, int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
void selectionSort(int *arr, int n)
{
    int indexOfMin;
    for (int i = 0; i < n - 1; i++)
    {
        indexOfMin = i;
        int j = i + 1;
        for (j; j < n; j++)
        {
            if (arr[j] < arr[indexOfMin])
            {
                indexOfMin = j;
            }
        }
        swap(&arr[i], &arr[indexOfMin]);
    }
}
int main()
{
    int arr[] = {7, 3, 4, 2, 8, 5, 65, 32, 6};
    printf("Before sorting : ");
```

```
    printArray(arr, 9);
    printf("After sorting : ");
    selectionSort(arr, 9);
    printArray(arr, 9);
    return 0;
}
```

Output:

```
Output                                                    Clear
/tmp/h2B8Zywvcs.o
Before sorting : 7 3 4 2 8 5 65 32 6
After sorting : 2 3 4 5 6 7 8 32 65
```

Question 2: Write a program to implement merge sort using array as a data structure.

Source Code:

```
#include <stdio.h>
void printArray(int *A, int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("%d ", A[i]);
    }
    printf("\n");
}
void merge(int A[], int mid, int low, int high)
{
    int i, j, k, B[100];
    i = low;
    j = mid + 1;
    k = low;
    while (i <= mid && j <= high)
    {
        if (A[i] < A[j])
        {
            B[k] = A[i];
            i++;
            k++;
        }
        else
        {
            B[k] = A[j];
            j++;
            k++;
        }
    }
    while (i <= mid)
    {
        B[k] = A[i];
        i++;
        k++;
    }
    while (j <= high)
```

```
    {
        B[k] = A[j];
        k++;
        j++;
    }
    for (int i = low; i <= high; i++)
    {
        A[i] = B[i];
    }
}
void mergeSort(int A[], int low, int high)
{
    int mid;
    if (low < high)
    {
        mid = (low + high) / 2;
        mergeSort(A, low, mid);
        mergeSort(A, mid + 1, high);
        merge(A, mid, low, high);
    }
}
int main()
{
    int a[] = {12, 31, 25, 8, 32, 17, 40, 42, 4, 3, 5, 7, 8, 96, 3, 2, 6, 89, 77};
    int n = sizeof(a) / sizeof(a[0]);
    printf("Before sorting : ");
    printArray(a, n);
    mergeSort(a, 0, n - 1);
    printf("After sorting : ");
    printArray(a, n);
    return 0;
}
```

Output:

```
Output                                                    Clear
/tmp/9OAfJWHOX8.o
Before sorting : 12 31 25 8 32 17 40 42 4 3 5 7 8 96 3 2 6 89 77
After sorting : 2 3 3 4 5 6 7 8 8 12 17 25 31 32 40 42 77 89 96
```

LAB-9

Question 1: Write a program to implement quick sort using array as a data structure.

Source Code:

```
#include <stdio.h>
void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
void printArray(int arr[], int size)
{
    for (int i = 0; i < size; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int partition(int array[], int low, int high)
{
    int pivot = array[high];

    int i = (low - 1);
    for (int j = low; j < high; j++)
    {
        if (array[j] <= pivot)
        {
            i++;

            swap(&array[i], &array[j]);
        }
    }
    swap(&array[i + 1], &array[high]);
    return (i + 1);
}
void quickSort(int arr[], int low, int high)
{
    int partitionIndex;
    if (low < high)
```

```
    {
        partitionIndex = partition(arr, low, high);
        quickSort(arr, low, partitionIndex - 1);
        quickSort(arr, partitionIndex + 1, high);
    }
}
int main()
{
    int arr[] = {3, 5, 2, 13, 12, 32, 42, 23, 31, 24, 35, 5};
    int length = 12;
    printf("Before Sorting Array : ");
    printArray(arr, length);
    printf("After Sorting Array : ");
    quickSort(arr, 0, length - 1);
    printArray(arr, length);
    return 0;
}
```

Output:

```
Output                                                    Clear
/tmp/h2B8Zywvcs.o
Before Sorting Array : 3 5 2 13 12 32 42 23 31 24 35 5
After Sorting Array : 2 3 5 5 12 13 23 24 31 32 35 42
```

Question 2: Write a program to implement bubble sort using array as a data structure.

Source Code:

```
#include <stdio.h>
void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
void printArray(int arr[], int size)
{
    for (int i = 0; i < size; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
void bubbleSort(int arr[], int size)
{
    for (int i = 0; i < size - 1; i++)
    {
        for (int j = 0; j < size - i - 1; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                swap(&arr[j], &arr[j + 1]);
            }
        }
    }
}
int main()
{
    int arr[] = {32, 53, 3, 143, 102};
    int length = 5;
    printf("Before Sorting Array : ");
    printArray(arr, length);
    printf("After Sorting Array : ");
    bubbleSort(arr, length);
    printArray(arr, length);
```

```
    return 0;
}
```

Output:

```
Output                                                    Clear
/tmp/h2B8Zywvcs.o
Before Sorting Array : 32 53 3 143 102
After Sorting Array : 3 32 53 102 143
```

LAB-10

Question 1: Write a program to implement Radix sort using array as a data structure.

Source Code:

```
#include <stdio.h>

int getMax(int arr[], int n)
{
    int max = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] > max)
        {
            max = arr[i];
        }
    return max;
}
void countSort(int arr[], int n, int exp)
{
    int output[n];
    int i, count[10] = {0};
    for (i = 0; i < n; i++)
    {
        count[(arr[i] / exp) % 10]++;
    }
    for (i = 1; i < 10; i++)
    {
        count[i] += count[i - 1];
    }
    for (i = n - 1; i >= 0; i--)
    {
        output[count[(arr[i] / exp) % 10] - 1] = arr[i];
        count[(arr[i] / exp) % 10]--;
    }
    for (i = 0; i < n; i++)
    {
        arr[i] = output[i];
    }
}
void radixsort(int arr[], int n)
{
```

```
    int max = getMax(arr, n);
    for (int exp = 1; max / exp > 0; exp *= 10)
        countSort(arr, n, exp);
}
void printArray(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
int main()
{
    int arr[] = {3, 5, 2, 13, 12};
    int length = 5;
    printf("Before Sorting Array : ");
    printArray(arr, length);
    printf("After Sorting Array : ");
    radixsort(arr, length);
    printArray(arr, length);
    return 0;
}
```

Output:

```
Output                                                    Clear
/tmp/h2B8Zywvcs.o
Before Sorting Array : 3 5 2 13 12
After Sorting Array : 2 3 5 12 13
```