HEMANG SINGH
1BM19CS061

~~KAX~~-XB

Practice - 3

1) Multiple Queue using 2D - array

```
# include<stdio.h>
# include <conio.h>
# include < stdlib.h>
# define  N  3
  int   queue [3][N];
  int   front [3] = { 0,0,0 };
  int   rear [ 3] = { -1, -1, -1 } ;
  int  item, pr;

  void  pqinsert (int pr) {
      if ( rear [pr] ==N -1)
         printf ("\n Queue overflow \n");
         else ;
        {
           printf (" \n ente the item \n");
           scanf (" %d", & item );
           rear[pq] ++;
           queue [pr] [ rear [pr]] = item ;
        }
        return ;
     }
  void pqdelete () {
        int i;
        for(i=0; i<3; i++) {
            if( rear [i] == front[i] -1)
                printf("\n Queue Empty \n");
            else {
                printf("deleted item is %d of queue %d
                                        \n", queue[i][front[i]], i+1);

                front[i]++;

                return;

            }
        }
}
void display () {
    int i, j;
    for( i=0; i<3; i++) {
        if( rear[i] == front[i] -1)
            printf("\n Queue empty %d \n", i+1);
        else {
            printf("\n Queue %d : ", i+1);
            for( j = front[i]; j<= rear[i]; j++)
                printf(" %d \t", queue[i][j]);

        }

}
 return;
}
void main () {

   int ch;
   while (1) {

       printf(" Priority Queue Operations\n");
printf(" \n 1. PQinsert \n2. PQdelete \n3.PQdisplay
        \n 4. Exit ");
Scanf(" %d ", &ch);
switch (ch)
 {
    Case 1: printf(" \nEnter the Priority
                       number \n");
            scanf(" %d", & pr);
            if(pr>0  && pr<4)
             pqinsert ( pr-1)
            else
               printf(" \n only  3 priority
                       exists  1, 2, 3 \n");

             break;
   Case 2: pqdelete ();
           break;
   Case 3: ~~pq~~ display();
           break;
   Case 4: exit(0) ;

 }
}
}
```

HEMANG SINGH
1BM19CS061

Practice - 3

Q) Ascending & Descending Priority Queue

```
# include < stdio.h>
# include < stdlib.h>
# define SIZE 5
int ~~PQinsert~~ PQ[SIZE], f=0; r=-1;

int PQinsert (int elem)
  {
   int i;
     if (Qfull())
     printf("\n Overflow !! \n");
     else
       {
        i = r;
        ++r;
        while( PQ[i] > =elem  && i>=0)
          {
           PQ[i+1] = PQ[i];
           i--;
          }
          PQ[i+1] = elem;

     }
  }
 int PQdelete - ASC()
   {
     int elem;
     if( Qempty()){
        printf ("\n Underflow !! \n");
        return (-1);
     }
else {
      elem  = PQ[f];
      f = f +1;
      return(elem);
      }
}
int   PQdelete_DES ()
   {
     int  elem;
     if ( Qempty ()) {
          printf (" \n Underflow !! \n");
          return(-1);
     }
      else
       {
       f = f +1;
        elem  = PQ[SIZE -f];
        return (elem);
     }
}
int  Qfull ()
    {
        if ( r = =  SIZE -1)
        return 1;
        return  0;
    }
int    Qempty ()
    {
      if ( r > r)
      return 1;
      return 0;
    }
```

```
Void display_ASC()
  {
    int i;
    if(Qempty())
      printf("\n Empty Queue \n");
      else
       {
        for(i=f ; i<=r; i++)
         printf("%d \t", PQ[i]);

       }

  }

Void display_DESC(){
    int i;
    if ( Qempty ())
    printf(" \n Empty Queue\n");
    else
     {
      for (i = SIZE -(f+1); i>=0;i--);
      printf("%d \t", PQ[i]);

    }

 }
Void main () {
   int  opn, ch, elem;
   printf(" Enter what you want : \n 1. Ascending
          priority Queue \n 2. Descending Priority Queue");
   Scanf(" %d", &ch);

   do
     {
       printf(" \n *** Priority Queue Operation **\n")
printf(" \n 1-Insert \n 2-Delete \n 3- Display \n
                                   4- Exit \n");

scanf ("%d", &opn);

switch (opn)
  {
    Case 1: printf("\n Enter element to be
                                Inserted \n");

            scanf("%d", & elem);
            PQinsert(elem);
            break;

    Case 2: if (ch == 1) {
            elem = PQdelete_ASC ();
            if(elem != -1)
              printf(" \n Deleted Element is:
                            %d \n", elem);

              break;
            }
            if (ch == 2) {
               elem = PQdelete_DES();
               if( elem != -1)
                 printf(" \n Deleted element
                      is : %d \n", elem);

               break;

    Case 3: if (ch == 1) {
            printf("\n Queue \n");
            display_ASC();
            break
          if( ch == 2) {
            printf(" \n Queue : \n");
            display_DESC();
            break;
            }

    Case 4: exit(0);
    default : printf(" \n Invalid Option !! ");
              break;

    }
   } while (opn != 4);
   }
```