Date ___/___/___

Q. Ascending Priority Queue

```
# include <stdio.h>
# include<stdlib.h>
# define QUE-SIZE 5
int item, rear = -1 , q [QUE_SIZE], count = 0;
void insertrear()
{
    if (rear = = QUESIZE - 1)
    {
    printf ("Queue overflow \n");
    return;
    }
    rear = rear+1;
    q [rear] = item ;
    count ++;
}
int deleteasc()
{
    int small = 99;
    int spos = -1;
    if (count ==0)
    {
        return -1;
    }
    for (int i =0 ; i< QUE_SIZE; i++)
    {
        if (q[i] <small)
        {
        small = q[i];
        spos = i;
        }
    }
```

Scanned with CamScanner

```
  q[spos] = 99;
  count = count -1;
  return small;
}
void display()
{
  int i;
  if(count = =0)
   {
    printf ("Queue empty \n");
    return;
   }
  printf ("Contents of queue \n");
  for (i=0; i< QUE_SIZE ; i++)
   {
     if q[i] = =99
      continue;
     else
     printf ("%d", q[i]);
   }
}
void main()
{
 int choice;
 for (;;)
{
 printf ("\n Enter 1. insertrear 2. delete 3. display\n");
 Scanf ("%d", & choice);
 Switch (choice)
 {
 case 1: printf ("Enter the item \n");
         Scanf ("%d" , & item);
```

Date ___ / ___ / ______

```
                insertrear ();
                break;
Case 2 :  item = deleteasc ();
          if (item == -1)
          printf ("Queue empty \n");
          else
          printf ("Item deleted is %d ", item);
          break;
Case 3 :  display ();
          break;
default : exit (0);
  }
}
```

Q DESCENDING Priority Queue

```
#include<stdio.h>
#include <stdlib.h>
#define QUE_SIZE 5
int item, rear=-1, q[QUE_SIZE], count=0;
void insertrear()
 {
    if (rear==QUE_SIZE-1)
     {
 printf ~~return~~ ("Queue overflow \n");
       return;
     }
     rear = rear + 1;
     q[rear] = item;
     count++;
 }
 int deletedesc()
  {
    int largest =0;
    int spos =-1;
    if (count ==0)
      {
       return -1;
      }
    for (int i=0; i<QUE_SIZE; i++)
      {
        if (q[i] > largest)
         {
          largest = q[i];
          spos = i;
         }
      }
```

Page No.

Date ___/___/_____

```
 q[spos] =0;
 count = count -1;
 return largest;
}
void display()
 {
   int i;
   if (count ==0)
   {
     printf ("Queue empty \n");
     return;
   }
   printf("Contents of queue \n");
  for(i=0; i<QUE_SIZE ;i++)
   {
     if (q[i] ==0)
       continue;
     else
     printf ("%d", q[i]);
   }
}
void main ()
 {
   int choice;
   for(;;)
   {
     printf ("\n ENTER 1. insertrear 2. delete
                                   3. display \n");
     scanf("%d", &choice);
    switch(choice)
   {
```

Date ___/___/_____

```
case 1:  printf ("Enter the item \n");
         scanf ("%d", & item)
         insertrear ();
        break;
 case 2:  item = deletefdesc ();
          if ( item = = -1)
          printf ("Queue empty \n");
          else
          printf ("Item deleted is %d", item);
         break();
case 3 :  display ();
          break;
default : exit (0);
}
  }
}
```

MULTIPLE PRIORITY QUEUE

```
#~~d~~ include <stdio.h>
# include <stdlib.h>
# define N 3
int queue [3][N];
int front [3]= {0, 0, 0};
int item, pr;
void main()
{
int ch;
while(1)
 {
  printf ("PRIORITY QUEUE");
 printf ("\n1:PQ insert");
 printf ("\n 2: PQ delete");
printf ("\n 3: PQ display ");
printf  ("\n 4 : Exit");
printf ("\n Enter the choice \n");
scanf ("%d", &ch);
 switch (ch)
{
 case 1: printf ("Enter the priority number \n");
         scanf ("%d", &pr);
         if (pr>0 && pr<4)
         pqinsert (pr-1);
         else
         printf ("only 3 priorty exists 1 2 3\n");
         break;
case 2: pq delete();
        break;
case3: display();
       break;
case 4: exit(0);
 }
}
}
pq insert(int pr)
{
 if(rear[pr]==N-1)
 printf("Queue Overflow\n");
 else
 printf("enter the item\n");
 scanf("%d", & item);
 rear[pr]++;
  queue[pr][rear[pr]]= item;
 }
return;
}
pq delete()
 {
   int i;
   for(i=0; i<3; i++)
    {
     if(rear[i] == front[i]-1)
     printf("Queue empty\n");
     else
      {
       printf("deleted item is %d of queue %d\n",
              queue[i] front[i], i+1);
       front[i]++;
        return;
      }
   } }
```

Page No.

Date ___/___/___

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

Scanned with CamScanner