Lab 6 - Priority Queue.

algorithm

```
insert()
{
  if (r = size - 1)
    print "Queue overflow"
  r = r + 1
  size[r] = item
    count++
}
```

```
display()
  if (f > r) {
    print "Queue is empty"
  }
  else
    print contents of que
```

```
sort()
 for (i=1; i< count; i++)
 {
    key = size[i]
    j = i-1
while
   (j >=0 AND size[j] > key)
      size [j+1] = size [j]
          j = j-1
 size[j+1] = key;
```

```
delete()
 if (f > r){
   f = 0      r = -1
       Queue is empty
 }
 print "Deleted", size[f++]
```

Scanned with CamScanner

program:

```
#include <stdio.h>
#include <stdlib.h>
# q-size 5
int r=-1, f=0, item, count=0;
int q[10], ch;
void insert_rear()
{
  if (r==q-size-1){
   printf("Queue overflow \n");
   return;
}
r=r+1;
q[r]=item;
count++;
}

void insertion_sort(){
  int i, j, key;
 for (i=1; i<count; i++)
 {
    key=q[i];
    j=i-1;
   while (j>=0 && q[j]>key){
      q[j+1]=q[j];
       j=j-1;
   }
   q[j+1]=key;
}
}
```

Scanned with CamScanner

```
void delete_front() {
    if (f > r) {
        f = 0;
        r = -1;
    printf("Queue is empty \n");
    return;
}
  print ("Item deleted = %d \n", q[f++]);
}
void display() {
     if (f > r) {
     printf ("Queue is empty\n");
     return;
}
  printf("Contents of the queue are : \n");
   for (int i=f ; i <r ; i++)
 {
     printf ("%d \n", q[i]);
  }
}
int main() {
 for (;;)
{
  printf("\n 1: Insert_rear \n 2. delete_front \n
          3: display \n");
  printf ("Enter the choice : \n");
  scanf("%d ", &ch);
 switch (ch) {
  case 1 : printf ("Enter the item : \n");
```

Scanned with CamScanner

```
            scanf("%d", &item);
            insert_rear();
            insertion_sort();
            break;
        case 2 : delete_front();
                 break;
        case 3 : display();
                 break;
        default : exit(0);
    }
}
    return 0;
}
```