04-11-2020

DS- Programs

Multiple priority queue program

```
# include <stdlib.h>
# include < stdio.h>
# define N 3
int queue [3] [N];
int front [3] = {0,0,0};
int rear [3] = {-1,-1,-1};
void pqinsert (int pr);
void pq ~~delete~~ ();
void ~~pq~~ display ();
int item , pr;
void item, pr;
void main()
{
int ch;
while (1)
{
printf ("PRIORITY QUEUE \n");
printf ("****** \n");
printf ("\n~~#~~1. PQinsert \n");
printf ("\n\t2. PQdelete \n");
printf (" n\t3. PQdisplay \n");
printf (" n\t4. Exit \n");
printf (" \nenter the choice \n");
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:  display();
           break;

  case 4: exit(0);
   }
  }
}
 void pqinsert (int pr)
  {
   if (rear[pr]==N-1)
   printf("\n Queue overflow \n");
   else
   {
    printf("\n Enter the item \n");
    Scanf ("%d", &item);
    rear [pr]++;
    queue [pr] [rear][pr]= item;
   }
    return;
   }
   void pq delete()
```

Scanned with CamScanner

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

2) Ascending program:-

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

#define MAX 3
int pq [MAX]
int count = 0;
int d > 0;
void insert (int data) {
 int i = 0;
 if (count = = MAX)
{
 printf ( " Queue overflow \n" );
 return;
 }
 if (count == 0) {
 pq [count ++] = data;
 } else {
   for (i= count -1; i>o= 0; i--) {
  if (data < pq [i]) {
  pq [i+1] = pq [i];
  } else {
  break;
 }
}
```

```
pq [i+1] = data;
 count ++;

}

}
int remove Data () {
 return pq [d++];
 }
 void display ()
 { int i;
 if (count == 0)
  {
   printf ("queue is empty \n");
    return;
}
 printf ("contents of queue: ");
 for (i = d ; i < count ; i++)
 {
   printf ("%d ", pq [i]);
 }
 printf ("\n");

 }
 int main () {
  int choic, item;
   for (;;)
   {
    printf ("\n1: insert 2: delete-smallest 3: display 4: exit \n");
    printf ("enter the choice : ");
    scanf ("%d", &choice );
    switch (choice)
```

Scanned with CamScanner

```
{
case 1: printf ("Enter the item to be inserted :");
scanf ("%d", &item);
 insert (item);
 break;
 case 2: item= removeData ();
  if (item= = -1)
  printf (" Queue is empty\n");
   else
  printf ("item deleted =%d\n", item);
   break;
   case3: display();
   break;
   default : exit(0);
 }
 }
}
```

Scanned with CamScanner

3) Descending_Order program

```
#include <stdio.h>
#include <stdlib.h>
#define 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;
   }
```

```
    }
}
void delete_rear (){
        if (f>r){
              f=0;
              r=-1;
              printf ("Queue is empty\n");
              return;
        }
      printf ("Item deleted =%d\n", q[r--]);
    }
    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 ("\n1: insert-rear \n2: delete-front
                  \n3: display \n");
```

```
printf ("Enter the choice :\n");
scanf ("%d", &ch);
switch (ch){

 case 1: printf (" Enter the item: \n");
  scanf ("%d", &item);
  insert_rear();
  insert_sort();
  break;
case 2: delete_rear();
   break;

 case3: display();
     break;
   default : exit(0);
  }
 }
 return 0;

}
```