```
x include < stdio. h>
* include < stdio h7
  typedef struct node
  void & data Ptr;
  struct node * next:
  } Queue _ Node;
   typedef struct
   QUEUE_NODE * front;
   QUEUE_NODE* tear ;
   Int count
   3 QUEUE
   QUEUE * create Queue (vois);
    bool engacue (QUEUE * queue, void * item Pty);
    Void print Queue (QUEUE* stack);
    int main (Void)
    11 Local Definitions
    QUEUE * queve 1;
    QUEUE* queue ?:
    QUEUE * queve 3:
     int * numptr;
     int * * item Ptr:
     queue 1 z create Queue ();
     queue ? = Create Queue ();
     queue 3 , Croute Queue V;
     int i = 9 ;
```

```
num Ptr = lint * ) malloc (size of 1i));
* number 21:
enquene (queue 1, numptr);
i = 8 ;
 num.Ptr = (Int *) malloc (sizeof (i));
* numptr z i.
 enqueue (queue1, numptr);
 iz 7;
  humPtrz (int*) malloc (sizeof (i));
  * num Ptr = i:
  enqueue (queue 1, numttr);
  Ĭ c 6;
 num Ptr = (int *) malloc (sizeof(i));
   * numptr = i.
   enqueue (queue ?, num Ptr);
   125;
    numPtr = (int *) malloc (sizeof(i));
   * numptr = i.
    enqueue (queue?, numPtr);
    iz4 ;
     numPtr = (int ") malloc (size of (1));
     * numptrzi.
     enqueue ( aueue ?, numptr );
     1 2 3;
      num Ptr = ( int *) mallog (size of (i));
     * number = i :
     enqueue (queueB. numptr) "
      numPtrz (int *) mahoc(sizeof(i));
    * num Prt zi;
```

```
if (queue =7 connt = = 0)
 queue - front = neaPtr;
 else
 queue -> rear -> next = how Ptr.
(queue -> eount) ++;
 queue - reur z nemper:
 return true ;
  Queue_ NODE * de letePtr;
  (gueup) fi
 while (queue -> front ! = NULL)
 tree (queue - 7 front -7 data Ptr):
 deleteftr = queue - 1 front;
  queue - tront z quieue - tront - Thext;
  free (deletePtr);
  tree (queue);
   teturn NULL.
   vold print Queue ( QUEVE* queae)
   QUEUE : MODE * node = queue -> front;
   Printf ("front =>");
   while (node)
   printf ("" bd", * [int + ) node -> data Ptr );
  hode = node -> next;
```

enqueus (queue 3, ni	mptr);	
ĭz1;	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
numptr = (Int +) ma	oc (size of (i));	
* humPrt zi.	The second section of the sect	and the second
enqueue (queue 3,	namptr);	
grintf 1' Queue 1: \		
print Queve (quare 1		/et 2015, edite:
print (" Queue 2:	(n');	
Print Queue (queue ?		
printaueue (queue		
print Queue (queue	3) .	
return o:	i i i i i i i i i i i i i i i i i i i	
3		
QUEUE * create C	u eue (Void)	
ζ.	*-0.5	170101
QUEUE * queue	Aughan Autoritation	
queue = CQUEUE *) malloc (size (QUEUE));	
if (queve)		
{		
dushe -> trout = NO	ч;	
queue -7 +car = NU	l;	
queue -7 count z	;	
}	- I wrote to be a little	
return queve;		
}	The state of the state of	
Lool enqueue (QUE	JE * Gueue, vold * itemptr)	ARCHIOL C
QUEUE NODE * no	uptr zi QUEUFNODE) malloc(sizeof(QU	EUR-NODE));
newPtr -> dataPtr = 1		
newPtr-7 next = N		
12/11/11/12/		

	NO :
7	
}	
built (, cs bear / u ,);	
return o	
}	
3	