Queues – Array Implementation

```
#include <stdio.h>
#define SIZE 10
#define EMPTY -1

typedef struct queue {
    int* elements;
    int front;
    int numEls;
    int size;
} queue;

void init(queue* q) {
  q->elements =
     malloc(sizeof(int)*SIZE);
  q->front = 0;
  q->numEls = 0;
  q->size = SIZE;
}

int enqueue(queue* q, int val) {
   int i;
   if (q->numEls != q->size) {
      int idx =(q->front +
                q->numEls)%q->size;

      q->elements[idx] = val;
      (q->numEls)++;

        return 1;
    }
      else { // Queue is full
        return 0;
    }
}

int empty(queue* q) {


}
```

```
int dequeue(queue* q) {
  if (q->numEls == 0)
        return EMPTY;

  int retval = q->elements[q->front];
  q->front = (q->front + 1)% q->size;
  (q->numEls)--;

  return retval;
}

int front(queue* q) {




}

int main() {
  queue* Q1 = malloc(sizeof(queue));
  init(Q1);

  queue* Q2 = malloc(sizeof(queue));
  init(Q2);

  enqueue(Q1, 8);
  enqueue(Q1, 3);
  enqueue(Q1, 6);
  enqueue(Q1, 7);
  enqueue(Q1, 9);
  for(i = 0; i < 6; i++){
     enqueue(Q2, dequeue(Q1));
     enqueue(Q1, i);
  }
// Point A

for(i = 0; i < 4; i++)
     dequeue(Q1);

// Point B
```

Show the contents of Q1 and Q2 after the code above is executed:

Point A:

Q1: front: _____ numElements: _____

elements:										

Q2: front: _____ numElements: _____

elements:										

Point B:

Q1: front: _____ numElements: _____

elements:										

Q2: front: _____ numElements: _____

elements:										

Queues – Linked List Implementation

```
#include <stdio.h>
#define EMPTY -1

typedef struct node {
    int data;
    struct node* next;
} node;

typedef struct queue {
    struct node* front;
    struct node* back;
} queue;

void init(queue* qPtr) {
     qPtr->front = NULL;
     qPtr->back = NULL;
}

int enqueue(queue* qPtr, int val) {
    node* temp =
(node*)malloc(sizeof(node));

    if (temp != NULL) {
        temp->data = val;
        temp->next = NULL;

        if (qPtr->back != NULL)
            qPtr->back->next = temp;

        qPtr->back = temp;

        if (qPtr->front == NULL)
            qPtr->front = temp;

        return 1;
    }
    else
        return 0;
}

int empty(queue* qPtr) {



}

int front(queue* qPtr) {




}
```

```
int dequeue(queue* qPtr) {
    if (qPtr->front == NULL)
        return EMPTY;

    int retval = qPtr->front->data;
    node *tmp = qPtr->front;
    qPtr->front = qPtr->front->next;

    if (qPtr->front == NULL)
        qPtr->back = NULL;

    free(tmp);

    return retval;
}

int main() {
    queue* MyQ = malloc(sizeof(queue));
    init(MyQ);

    enqueue(MyQ, 3);
    enqueue(MyQ, 7);
    enqueue(MyQ, 4);

    printf("Dequeue %d\n",
            dequeue(MyQ));

    enqueue(MyQ, 2);
    printf("Dequeue %d\n",
            dequeue(MyQ));
    printf("Front of Queue: %d\n",
            front(MyQ));
    printf("Dequeue %d\n",
            dequeue(MyQ));
    printf("Dequeue %d\n",
            dequeue(MyQ));

    printf("Is empty: %d\n",
            empty(MyQ));

    enqueue(MyQ, 5);
    enqueue(MyQ, 9);
    printf("Dequeue %d\n",
            dequeue(MyQ));
    printf("Dequeue %d\n",
            dequeue(MyQ));

    return 0;
}
```