```c
//*****************************************************************************
//  List.c
//
//  List and queue utility functions.
//
//  Leonardo C. Monteiro (leonardo.monteiro.lcm@gmail.com)
//  https://www.linkedin.com/in/leonardo-monteiro-950112345/
//*****************************************************************************

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <cstddef>
#include "List.h"

#define DATA_BUFFER_PAGE 1000

typedef struct ListRec
{
	DATA         *buf;
	unsigned int  nbuf;
	unsigned int  nlist;
} LISTREC, *PLISTREC;

//******************************
// List functions
//******************************

#define _GetListRecord(list) ((PLISTREC)list)

LIST NewList()
{
	PLISTREC lst = malloc(sizeof(LISTREC));
	lst->buf = malloc(DATA_BUFFER_PAGE * sizeof(DATA));
	lst->nbuf = DATA_BUFFER_PAGE;
	lst->nlist = 0;
	return lst;
}

void FreeList(LIST list)
{
	PLISTREC lst = _GetListRecord(list);
	free(lst->buf);
	free(lst);
}

unsigned int ListCount(LIST list)
{
	PLISTREC lst = _GetListRecord(list);
	return lst->nlist;
}

DATA ListAt(LIST list, unsigned int index)
{
	PLISTREC lst = _GetListRecord(list);
	return (index < lst->nlist ? lst->buf[index] : NULL);
}

DATA ListHead(LIST list)
{
	PLISTREC lst = _GetListRecord(list);
	return (lst->nlist > 0 ? lst->buf[0] : NULL);
}

DATA ListTail(LIST list)
{
	PLISTREC lst = _GetListRecord(list);
	return (lst->nlist > 0 ? lst->buf[lst->nlist - 1] : NULL);
}

void ListPushTail(LIST list, DATA data)
{
	PLISTREC lst = _GetListRecord(list);
	lst->nlist += 1;
	if (lst->nlist > lst->nbuf)
	{
		lst->nbuf += DATA_BUFFER_PAGE;
		lst->buf = realloc(lst->buf, lst->nbuf * sizeof(DATA));
	}
	lst->buf[lst->nlist - 1] = data;
}

DATA ListPopHead(LIST list)
{
	PLISTREC lst = _GetListRecord(list);
	DATA data = NULL;
	if (lst->nlist > 0)
	{
		data = lst->buf[0];
		lst->nlist -= 1;
		for (unsigned int i = 0; i < lst->nlist; i++)
			lst->buf[i] = lst->buf[i + 1];
	}
	return data;
}

//********************************
// Queue functions
//********************************

QUEUE NewQueue()
{
	return NewList();
}

void FreeQueue(QUEUE queue)
{
	FreeList(queue);
}

unsigned int QueueCount(QUEUE queue)
{
	return ListCount(queue);
}

DATA QueueAt(QUEUE queue, unsigned int index)
{
	return ListAt(queue, index);
}

DATA QueueHead(QUEUE queue)
{
	return ListHead(queue);
}

DATA QueueTail(QUEUE queue)
{
	return ListTail(queue);
}
```

```c
void QueuePush(QUEUE queue, DATA data)
{
	ListPushTail(queue, data);
}

DATA QueuePop(QUEUE queue)
{
	return ListPopHead(queue);
}
```