Stacks and Queues (+READMEs, GDB, Makefiles)

GDB:

- This is a debugger
  - There are many different debuggers-your IDE may have a debugger
  - Codeblocks: https://www.youtube.com/watch?v=IN_RTt_5cf0
- You can use it (along with Valgrind) to help you find bugs in your program
- You will see me use different commands in gdb-note there are many more (you can look them up online)
- I will be showing a few small examples today

EXAMPLE 1: lights1.c

```
[fiq8745@omega ~]$ gcc -g lights1.c -o lights /*(use –g to help with debugging) it compiles fine…*/
[fiq8745@omega ~]$ ./lights lightstuff1.txt
Segmentation fault  /*but it crashes at run time*/
[fiq8745@omega ~]$ gdb lights
GNU gdb (GDB) Red Hat Enterprise Linux (7.0.1-45.el5)
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/f/fi/fiq8745/lights...done.
(gdb) r lightstuff1.txt /*running the program in gdb-don't forget to include your file*/
Starting program: /home/f/fi/fiq8745/lights lightstuff1.txt
warning: no loadable sections found in added symbol-file system-supplied DSO at 0x2aaaaaaab000

Program received signal SIGSEGV, Segmentation fault. /*problem here*/
0x00000000004009a6 in light_info (filename=0x7fffffffec89 "lightstuff1.txt") at lights1.c:101 /*line number of problem*/
101                     *(p->details[0])=atoi(token); /*line causing problem*/
(gdb)
```

(in class: show actual issue in code on Omega)

You can also see it on Valgrind:

Using Valgrind:

```
[fiq8745@omega ~]$ valgrind --tool=memcheck --leak-check=yes --show-reachable=yes --track-origins=yes --num-callers=20 --track-fds=yes ./lights lightstuff1.txt
==575== Memcheck, a memory error detector
==575== Copyright (C) 2002-2009, and GNU GPL'd, by Julian Seward et al.
==575== Using Valgrind-3.5.0 and LibVEX; rerun with -h for copyright info
==575== Command: ./lights lightstuff1.txt
==575==
==575== Use of uninitialised value of size 8
==575==    at 0x4009A6: light_info (lights1.c:101) /*tells you the line with the issue*/
```

```
==575==    by 0x400795: main (lights1.c:19)
==575==  Uninitialised value was created by a heap allocation /*what's wrong*/
==575==    at 0x4A0610C: malloc (vg_replace_malloc.c:195)
==575==    by 0x400962: light_info (lights1.c:86)
==575==    by 0x400795: main (lights1.c:19)
==575==
==575== Invalid read of size 8
==575==    at 0x4009A6: light_info (lights1.c:101)
==575==    by 0x400795: main (lights1.c:19)
==575==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==575==
==575==
==575== Process terminating with default action of signal 11 (SIGSEGV)
==575==  Access not within mapped region at address 0x0
==575==    at 0x4009A6: light_info (lights1.c:101)
==575==    by 0x400795: main (lights1.c:19)
==575==  If you believe this happened as a result of a stack
==575==  overflow in your program's main thread (unlikely but
==575==  possible), you can try to increase the size of the
==575==  main thread stack using the --main-stacksize= flag.
==575==  The main thread stack size used in this run was 10485760.
test==575==
==575== FILE DESCRIPTORS: 4 open at exit.
==575== Open file descriptor 3: lightstuff1.txt
==575==    at 0x3EC94C6AC0: __open_nocancel (in /lib64/libc-2.5.so)
==575==    by 0x3EC946B3F2: _IO_file_open (in /lib64/libc-2.5.so)
==575==    by 0x3EC946B53B: _IO_file_fopen@@GLIBC_2.2.5 (in /lib64/libc-2.5.so)
==575==    by 0x3EC9460763: __fopen_internal (in /lib64/libc-2.5.so)
==575==    by 0x4008DF: light_info (lights1.c:66)
==575==    by 0x400795: main (lights1.c:19)
==575==
==575== Open file descriptor 2: /dev/pts/31
==575==    <inherited from parent>
==575==
==575== Open file descriptor 1: /dev/pts/31
==575==    <inherited from parent>
==575==
==575== Open file descriptor 0: /dev/pts/31
==575==    <inherited from parent>
==575==
==575==
==575== HEAP SUMMARY:
==575==     in use at exit: 712 bytes in 4 blocks
==575==   total heap usage: 4 allocs, 0 frees, 712 bytes allocated
==575==
==575== 20 bytes in 1 blocks are still reachable in loss record 1 of 4
==575==    at 0x4A0610C: malloc (vg_replace_malloc.c:195)
==575==    by 0x400970: light_info (lights1.c:87)
==575==    by 0x400795: main (lights1.c:19)
==575==
==575== 24 bytes in 1 blocks are still reachable in loss record 2 of 4
==575==    at 0x4A0610C: malloc (vg_replace_malloc.c:195)
==575==    by 0x400962: light_info (lights1.c:86)
```

```
==575==    by 0x400795: main (lights1.c:19)
==575==
==575== 100 bytes in 1 blocks are still reachable in loss record 3 of 4
==575==    at 0x4A0610C: malloc (vg_replace_malloc.c:195)
==575==    by 0x400914: light_info (lights1.c:74)
==575==    by 0x400795: main (lights1.c:19)
==575==
==575== 568 bytes in 1 blocks are still reachable in loss record 4 of 4
==575==    at 0x4A0610C: malloc (vg_replace_malloc.c:195)
==575==    by 0x3EC9460709: __fopen_internal (in /lib64/libc-2.5.so)
==575==    by 0x4008DF: light_info (lights1.c:66)
==575==    by 0x400795: main (lights1.c:19)
==575==
==575== LEAK SUMMARY:
==575==    definitely lost: 0 bytes in 0 blocks
==575==    indirectly lost: 0 bytes in 0 blocks
==575==      possibly lost: 0 bytes in 0 blocks
==575==    still reachable: 712 bytes in 4 blocks
==575==         suppressed: 0 bytes in 0 blocks
==575==
==575== For counts of detected and suppressed errors, rerun with: -v
==575== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 4 from 4)
Segmentation fault
```

EXAMPLE 2: lights2.c

```
[fiq8745@omega ~]$ gcc lights2.c
[fiq8745@omega ~]$ ./a.out lightstuff1.txt
on
***Turning lights on:
bb b b bb bb b  /*it's running, but not doing what it's supposed to do (all letters are b)*/

exit
Exiting...

[fiq8745@omega ~]$ gcc -g lights2.c -o lights
[fiq8745@omega ~]$ gdb lights
GNU gdb (GDB) Red Hat Enterprise Linux (7.0.1-45.el5)
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/f/fi/fiq8745/lights...done.
(gdb) b main /*setting a break point at main-where I'm starting my step through of the program*/
Breakpoint 1 at 0x400a2d: file lights2.c, line 142.
(gdb) r lightstuff1.txt /*run the program step by step*/
Starting program: /home/f/fi/fiq8745/lights lightstuff1.txt
```

```
warning: no loadable sections found in added symbol-file system-supplied DSO at 0x2aaaaaaab000

Breakpoint 1, main (argc=2, argv=0x7fffffffea48) at lights2.c:142
142             Node* head = malloc(sizeof(Node));
(gdb) s       /*using s to step through each line of program*/
144             char *userInput=malloc(20);
(gdb) s
145             char  *test = "NULL";
(gdb)         /*here I'm just pushing enter instead of s to move through*/
147             if(argc <2)
(gdb) s
153             head = light_info(argv[1]);
(gdb)
light_info (filename=0x7fffffffec8a "lightstuff1.txt") at lights2.c:16
16              char *line=malloc(100);
(gdb)
18              Node *head = NULL;
(gdb)
19              Node *temp = NULL;
(gdb)
20              Node *linked=NULL;
(gdb)
23              fp=fopen(filename, "r");
(gdb)
25              if(fp == NULL)
(gdb)
30              while(fgets(line,100,fp)!=NULL)
(gdb)
32                      temp = malloc(sizeof(Node));
(gdb)
33                      temp->color=malloc(20);/*should be longer than 1 (size(char)==1)*/
(gdb)
34                      temp->color=strtok(line,","); /*you shouldn't be directly doing this...*/
 (gdb) quit

/*another round:*/


[fiq8745@omega ~]$ gdb lights
GNU gdb (GDB) Red Hat Enterprise Linux (7.0.1-45.el5)
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/f/fi/fiq8745/lights...done.
(gdb) b 30 /*I'm picking where to start stepping through the program (line)*/
Breakpoint 1 at 0x40089a: file lights2.c, line 30.
(gdb) r lightstuff1.txt
Starting program: /home/f/fi/fiq8745/lights lightstuff1.txt
```

```
warning: no loadable sections found in added symbol-file system-supplied DSO at 0x2aaaaaaab000

/*we're in a loop, this is the first iteration*/
Breakpoint 1, light_info (filename=0x7fffffffec89 "lightstuff1.txt") at lights2.c:30
30              while(fgets(line,100,fp)!=NULL) /*starting to step through my program at line 30 (my
breakpoint*/
(gdb) s
32                      temp = malloc(sizeof(Node));
(gdb) s
33                      temp->color=malloc(20);//should be longer than 1 (size(char)==1)*
(gdb) print temp        /*I can take a look at what the value of any variable is by saying print or p*/
$1 = (Node *) 0x602300
(gdb) p temp->color /*I can take a look at what the value of any variable is by saying print or p*/
$2 = 0x0
(gdb) s
34                      temp->color=strtok(line,","); //you shouldn't be directly doing this...
(gdb) p temp->color
$3 = 0x602320 ""
(gdb) s
43                      temp->details=malloc(sizeof(int*)*2); //you didnt malloc details
(gdb)
45                      temp->details[0] = malloc(sizeof(int));
(gdb)
46                      temp->details[1] = malloc(sizeof(int));
(gdb)
48                      *(temp->details[0]) = atoi(strtok(NULL,","));//bright
(gdb)
49                      *(temp->details[1]) = atoi(strtok(NULL,"\n"));//size
(gdb) print *(temp->details[0])
$4 = 2
(gdb) s
50                      temp->next = NULL;
(gdb) s
52                      if(head==NULL)
(gdb)
54                              head=temp;
(gdb)

/*we're in a loop, so we hit the breakpoint (line 30) again…this is the second iteration*/
Breakpoint 1, light_info (filename=0x7fffffffec89 "lightstuff1.txt") at lights2.c:30
30              while(fgets(line,100,fp)!=NULL)
(gdb) print head
$5 = (Node *) 0x602300
(gdb) print temp
$6 = (Node *) 0x602300
(gdb) print *head /*here I can see all the values of the struct pointed to by head*/
$7 = {color = 0x602050 "blue", details = 0x602340, next = 0x0}
(gdb) quit
```

Look at the comments in this code to see the actual issue that was happening in this code (it would be too long to actually debug the whole program in class)

EXAMPLE 3: decimalchop.c

Issue: why does the ascending order "chop off" the part after the decimal?

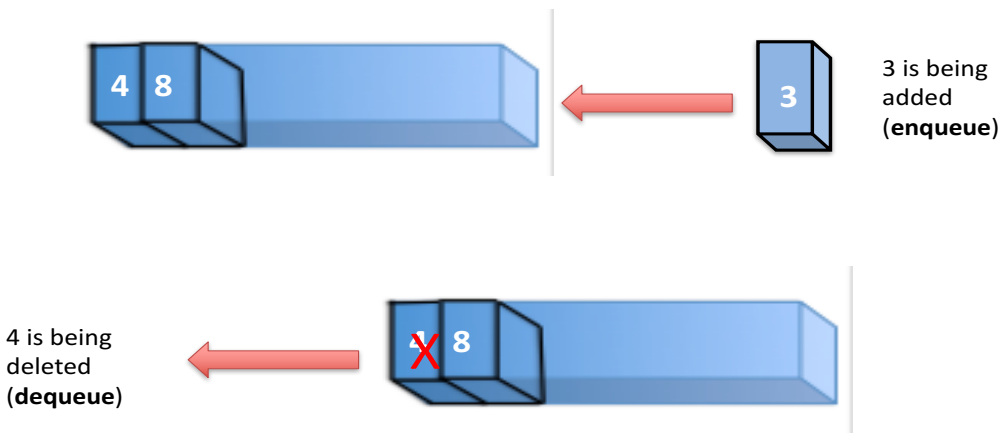
```
Enter total number of elements: 3

Enter Number 1: 6.1
Enter Number 2: 1.1
Enter Number 3: 4.1

After Sorting in Ascending Order:

1.00

4.00

6.10
```

```
[fiq8745@omega Programs]$ gcc -g decimalchop.c -o main
[fiq8745@omega Programs]$ gdb main
GNU gdb (GDB) Red Hat Enterprise Linux (7.0.1-45.el5)
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/f/fi/fiq8745/Programs/main...done.
(gdb) b 30 /*I picked the line number based on previous debugging*/
Breakpoint 1 at 0x400705: file decimalchop.c, line 30.
(gdb) s
The program is not being run.
(gdb) r /*need to start the program to step through it*/
Starting program: /home/f/fi/fiq8745/Programs/main
warning: no loadable sections found in added symbol-file system-supplied DSO at 0x2aaaaaaab000
Enter total number of elements: 3 /*program runs normally until we get to breakpoint*/

Enter Number 1: 4.1
Enter Number 2: 5.1
Enter Number 3: 6.1

Breakpoint 1, main () at decimalchop.c:30  /*we hit the breakpoint, so now we start stepping through*/
30              for(j=0;j<=i;j++)
(gdb) s
32                  if(*(data+i)<*(data+j))
(gdb) s
30              for(j=0;j<=i;j++)
(gdb) s
28              for(i=0;i<num;i++)
(gdb) s

Breakpoint 1, main () at decimalchop.c:30
30              for(j=0;j<=i;j++)
```

```
(gdb)
32                  if(*(data+i)<*(data+j))
(gdb)
30              for(j=0;j<=i;j++)
(gdb)
32                  if(*(data+i)<*(data+j))
(gdb)
30              for(j=0;j<=i;j++)
(gdb)
28            for(i=0;i<num;i++)
(gdb)

Breakpoint 1, main () at decimalchop.c:30
30              for(j=0;j<=i;j++)
(gdb)
32                  if(*(data+i)<*(data+j))
(gdb)
30              for(j=0;j<=i;j++)
(gdb)
32                  if(*(data+i)<*(data+j))
(gdb)
30              for(j=0;j<=i;j++)
(gdb)
32                  if(*(data+i)<*(data+j))
(gdb)
30              for(j=0;j<=i;j++)
(gdb)
28            for(i=0;i<num;i++)
(gdb)
40            printf("\nAfter Sorting in Ascending Order: \n");
(gdb)

After Sorting in Ascending Order:
42            for(i=0;i<num;i++)
(gdb)
44              printf("\n%lf\n",*(data+i));
(gdb)

4.100000 /*what the user sees during the run of the program*/
42            for(i=0;i<num;i++)
(gdb)
44              printf("\n%lf\n",*(data+i));
(gdb)

5.100000
42            for(i=0;i<num;i++)
(gdb)
44              printf("\n%lf\n",*(data+i));
(gdb)

6.100000
42            for(i=0;i<num;i++)
(gdb)
```

```
48          for (i = 0; i < num; ++i)
(gdb)
50              for (j = i + 1; j < num; ++j)
(gdb)
52                  if (*(data+i) <* (data+j))
(gdb)
54                      t=*(data+i);
(gdb)
55                      *(data+i)=*(data+j);
(gdb) print *(data+i)  /*printing out variables*/
$1 = 4.0999999
(gdb) print t  /*printing out variables-notice it's different here.  Found our problem (t is an int instead of a float, so decimal portion not saved*/
$2 = 4
(gdb) quit
```

Queue:



-Insertion (enqueue) is ONLY allowed from the end of the queue
-Deletion (dequeue) is ONLY allowed from the front of the queue
-First in, first out (FIFO) (since 4 was the first in, it is the first out)

Stacks and Queues

Stack Operations:

- Push
- Pop
- Peek
- Empty?
- Full?

(There can be more or less)

Queue Operations:

- Enqueue
- Dequeue
- Peek
- Empty?
- Full?

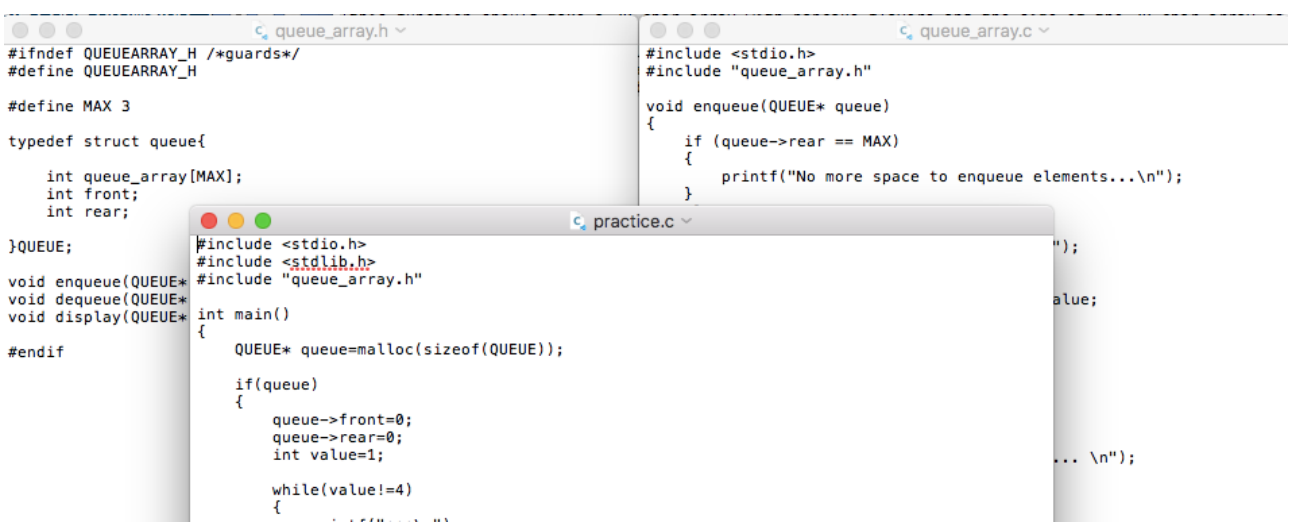
(There can be more or less)

NOTE: I DID NOT FREE/CHECK IF MALLOC RETURED NULL WHEN I MALLOCED IN THESE EXAMPLES

Queue-Array implementation

```
(base) Computers-Air:C computer$ gcc practice.c queue_array.c
(base) Computers-Air:C computer$ ./a.out
***
1. Enqueue
2. Dequeue
3. Display
4. Quit
1
-Enter value to enqueue:
10
***
1. Enqueue
2. Dequeue
3. Display
4. Quit
1
-Enter value to enqueue:
20
***
1. Enqueue
2. Dequeue
3. Display
4. Quit
1
-Enter value to enqueue:
30
***
1. Enqueue
2. Dequeue
3. Display
4. Quit
1
```

```
No more space to enqueue elements...
***
1. Enqueue
2. Dequeue
3. Display
4. Quit
3
Queue is :
10 20 30
***
1. Enqueue
2. Dequeue
3. Display
4. Quit
2
Element dequeued from queue is: 10
***
1. Enqueue
2. Dequeue
3. Display
4. Quit
2
Element dequeued from queue is: 20
***
1. Enqueue
2. Dequeue
3. Display
4. Quit
3
Queue is :
30
***
1. Enqueue
2. Dequeue
3. Display
4. Quit
4
Exiting...
```



[queue_array.h](#)

```
#ifndef QUEUEARRAY_H /*guards*/
#define QUEUEARRAY_H

#define MAX 3

typedef struct queue{
```

```
   int queue_array[MAX];
   int front;
   int rear;

}QUEUE;

void enqueue(QUEUE* queue);
void dequeue(QUEUE* queue);
void display(QUEUE* queue);

#endif
```

queue_array.c

```
#include <stdio.h>
#include "queue_array.h"

void enqueue(QUEUE* queue)
{
   if (queue->rear == MAX)
   {
      printf("No more space to enqueue elements...\n");
   }
   else
   {   int value;
      printf("-Enter value to enqueue: \n");
      scanf("%d", &value);

      queue->queue_array[queue->rear] = value;
      queue->rear +=1;
   }
}

void dequeue(QUEUE* queue)
{
   if (queue->front == queue->rear)
   {
      printf("Nothing in queue to dequeue... \n");
      return ;
   }
   else
   {
      printf("Element dequeued from queue is: %d\n", queue->queue_array[queue->front]);
      queue->front += 1;
   }
}

void display(QUEUE* queue)
{
```

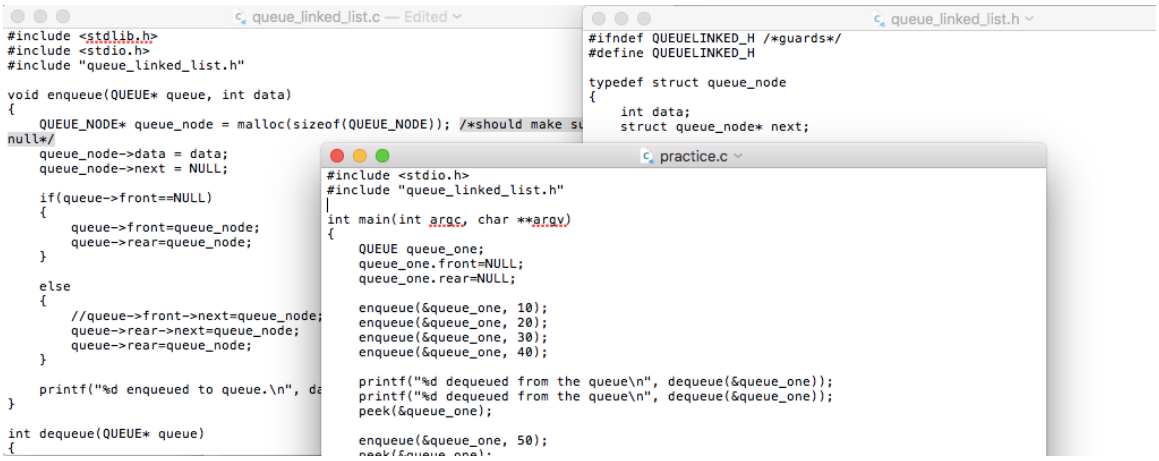
```
    int i;
    if (queue->rear == 0)
        printf("Queue is empty \n");
    else
    {
        printf("Queue is : \n");
        for (i = queue->front; i < queue->rear; i++)
        printf("%d ", queue->queue_array[i]);
        printf("\n");
    }
}
```

practice.c

```
#include <stdio.h>
#include <stdlib.h>
#include "queue_array.h"

int main(int argc, char** argv)
{
    QUEUE* queue=malloc(sizeof(QUEUE));

    if(queue)
    {
        queue->front=0;
        queue->rear=0;
        int value=1;

        while(value!=4)
        {
            printf("***\n");
            printf("1. Enqueue\n");
            printf("2. Dequeue\n");
            printf("3. Display\n");
            printf("4. Quit \n");
            scanf("%d", &value);

            if(value==1)
            {
                enqueue(queue);
            }

            else if(value==2)
            {
                dequeue(queue);
            }

            else if(value==3)
            {
                display(queue);
```

```
        }

        else if(value==4)
        {
            printf("Exiting...\n");
        }

        else
        {
            printf("Unknown response...\n");
        }

        if(queue->front==queue->rear)
        {
            queue->front=0;
            queue->rear=0;
        }
    }

    free(queue);
  }

}
```

Queue-Linked list implementation



queue_linked_list.c — Edited

```
#include <stdlib.h>
#include <stdio.h>
#include "queue_linked_list.h"

void enqueue(QUEUE* queue, int data)
{
    QUEUE_NODE* queue_node = malloc(sizeof(QUEUE_NODE)); /*should make su
null*/
    queue_node->data = data;
    queue_node->next = NULL;

    if(queue->front==NULL)
    {
        queue->front=queue_node;
        queue->rear=queue_node;
    }

    else
    {
        //queue->front->next=queue_node;
        queue->rear->next=queue_node;
        queue->rear=queue_node;
    }

    printf("%d enqueued to queue.\n", da
}

int dequeue(QUEUE* queue)
{
```

queue_linked_list.h

```
#ifndef QUEUELINKED_H /*guards*/
#define QUEUELINKED_H

typedef struct queue_node
{
    int data;
    struct queue_node* next;
```

practice.c

```
#include <stdio.h>
#include "queue_linked_list.h"

int main(int argc, char **argv)
{
    QUEUE queue_one;
    queue_one.front=NULL;
    queue_one.rear=NULL;

    enqueue(&queue_one, 10);
    enqueue(&queue_one, 20);
    enqueue(&queue_one, 30);
    enqueue(&queue_one, 40);

    printf("%d dequeued from the queue\n", dequeue(&queue_one));
    printf("%d dequeued from the queue\n", dequeue(&queue_one));
    peek(&queue_one);

    enqueue(&queue_one, 50);
```

```
(base) Computers-Air:C computer$ gcc practice.c queue_linked_list.c
(base) Computers-Air:C computer$ ./a.out
10 enqueued to queue.
20 enqueued to queue.
30 enqueued to queue.
40 enqueued to queue.
10 dequeued from the queue
20 dequeued from the queue
front of the queue: 30, Rear of the queue: 40
50 enqueued to queue.
front of the queue: 30, Rear of the queue: 50
```

queue_linked_list.h

```
#ifndef QUEUELINKED_H /*guards*/
#define QUEUELINKED_H

typedef struct queue_node
{
   int data;
   struct queue_node* next;

}QUEUE_NODE;

typedef struct queue{
   QUEUE_NODE* front;
   QUEUE_NODE* rear;

}QUEUE;

void enqueue(QUEUE* queue, int data);
int dequeue(QUEUE* queue);
void peek(QUEUE* queue);

#endif
```

queue_linked_list.c

```
#include <stdlib.h>
#include <stdio.h>
#include "queue_linked_list.h"

void enqueue(QUEUE* queue, int data)
{
   QUEUE_NODE* queue_node = malloc(sizeof(QUEUE_NODE)); /*should make sure not null*/
   queue_node->data = data;
   queue_node->next = NULL;

   if(queue->front==NULL)
   {
      queue->front=queue_node;
      queue->rear=queue_node;
   }

   else
   {
      //queue->front->next=queue_node;
      queue->rear->next=queue_node;
      queue->rear=queue_node;
   }

   printf("%d enqueued to queue.\n", data);
}
```

```
int dequeue(QUEUE* queue)
{
    if(queue->front)
    {
        int value=queue->front->data;
        QUEUE_NODE* temp = queue->front;
        queue->front=queue->front->next;
        free(temp);
        return value;
    }

    else
    {
        printf("Queue is empty...\n");
        return -1;
    }

}

//need to keep track of current front of the queue...could also run a loop each time to get to the end
(front..first inserted)
void peek(QUEUE* queue)
{
    printf("front of the queue: %d, ", queue->front->data);
    printf("Rear of the queue: %d\n", queue->rear->data);
}
```

practice.c

```
#include <stdio.h>
#include "queue_linked_list.h"

int main(int argc, char **argv)
{
    QUEUE queue_one;
    queue_one.front=NULL;
    queue_one.rear=NULL;

    enqueue(&queue_one, 10);
    enqueue(&queue_one, 20);
    enqueue(&queue_one, 30);
    enqueue(&queue_one, 40);

    printf("%d dequeued from the queue\n", dequeue(&queue_one));
    printf("%d dequeued from the queue\n", dequeue(&queue_one));
    peek(&queue_one);

    enqueue(&queue_one, 50);
    peek(&queue_one);
```

```
  return 0;
}
```

Makefiles:

Ok, you probably noticed that to add a headerfile, we needed to compile two files on the command line. What if we create 3 or 4 header files? Then what? We would need to write them all out when compiling. What if we wanted to delete certain files after? Once again, I would manually do that. Is there a way to avoid this? Yes! We can use makefiles.

Documentation:
https://www.gnu.org/software/make/manual/html_node/Makefiles.html#Makefiles

I will go over the general idea of a makefile (see above link for more details). **Note that the examples I am giving are very SIMPLE examples.**

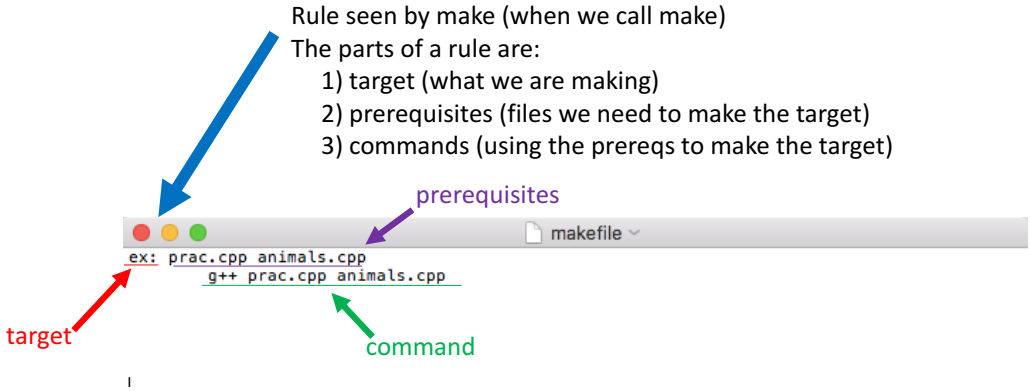
(some of the examples below are in C++…the idea is the same in C)

1) *make* is a command (called a utility). Just like *ls* is a command (to list files in a directory) we can use the *make* command to automate processes (like the process of putting together all pieces of an executable program together).

2) When we call *make* (saying we want to execute the *make* command), *make* knows what we want it do by reading the *makefile* that we create.

3) A *makefile* is a special file that contains commands of we want to do. It contains *rules* (it can also include other items) that we create (basically all the steps we want to automate). The basic structure of a rule in a *makefile* is:

Target: prerequisites
<TAB> Commands (recipes)



makefile

```
#THIS IS A VERY VERY SIMPLE MAKEFILE (# IS A COMMENT)
#YOU CAN PUT @ BEFORE A COMMAND SO IT DOESN'T SHOW UP ON SCREEN

executable:
	gcc -o main1 queue_array.c class17queueh1.c #COMMAND

clean:
	echo "Cleaning up executable!" #output to screen
	rm main1
```

We can compile the files like this (headerfile, main-no makefile):

```
computer$ g++ -std=c++11 main.cpp mouse.cpp
```

or we could create a makefile (# used for comments):

makefile

```
ex:	#TARGET
	g++ -std=c++11 main.cpp mouse.cpp  #COMMAND
```

When I run the makefile by typing make, it looks like this (the command called is shown):

```
mouse computer$ make
g++ -std=c++11 main.cpp mouse.cpp  #COMMAND
mouse computer$ ./a.out
Is this a house or school?
```

If I don't want the command to show, I could add @ at the beginning of my command:

makefile — Edited

```
ex:	#TARGET
	@g++ -std=c++11 main.cpp mouse.cpp  #COMMAND
```

Notice now the command doesn't show up when I type in make:

```
mouse computer$ make
Computers-MacBook-Air:mouse computer$ ./a.out
Is this a house or school?
```

We could also break up the compilation process by directly creating object files and then putting them together to make the executable (no makefile):

```
computer$ g++ -std=c++11 -c mouse.cpp  //making an object file
(mouse.o)
computer$ ls
main.cpp  mouse.cpp  mouse.h mouse.o //we see mouse.o
computer$ g++ -std=c++11 -c main.cpp //making an object file (main.o)
computer$ ls
```

```
main.cpp   main.o mouse.cpp mouse.h mouse.o //we see main.o
computer$ g++ -std=c++11 -o main1 main.o mouse.o //make an
executable from the object files (main1)
computer$ ls
main.cpp   main.o main1 mouse.cpp mouse.h mouse.o //we see main1
computer$ ./main1 //we can now run main1
```

Above, I created two object files (mouse.o and main.o). I then has them put them together into an executable called main1 (I could have also just used the default name a.out instead of giving it the name main1).

A *makefile* for the preceding would look like: (notice it's pretty much the same as the steps I took manually, just combined into a single file)

makefile — Edited

```
executable: main.o mouse.o   #TARGET, PREREQs (notice here the prereqs are targets made below)
        g++ -std=c++11 -o main1 main.o mouse.o #COMMAND

main.o: main.cpp #TARGET, PREREQs
        g++ -std=c++11 -c main.cpp #COMMAND

mouse.o: mouse.cpp #TARGET, PREREQs
        g++ -std=c++11 -c mouse.cpp #COMMAND
```

```
computer$ make
g++ -std=c++11 -c main.cpp #COMMAND
g++ -std=c++11 -c mouse.cpp #COMMAND
g++ -std=c++11 -o main1 main.o mouse.o #COMMAND
```

Remember you could put @ before each command so it doesn't show up:

makefile

```
executable: main.o mouse.o   #TARGET, PREREQs (notice here the prereqs are targets made below)
        @g++ -std=c++11 -o main1 main.o mouse.o #COMMAND

main.o: main.cpp #TARGET, PREREQs
        @g++ -std=c++11 -c main.cpp #COMMAND

mouse.o: mouse.cpp #TARGET, PREREQs
        @g++ -std=c++11 -c mouse.cpp #COMMAND
```

```
computer$ make
Computers-MacBook-Air:mouse computer$ ls
a.out      main.cpp   main.o          main1      makefile
     mouse.cpp  mouse.h         mouse.o
```

Above, notice when I type *ls*, the object files I created show up (mouse.o and main.o).

Sometimes we just want to execute a process and not make a target file. These are called *phony targets*. An example is when I want to "clean up" my files created (my object files and the executable). I can make a rule to do this.

Notice I am typing *make* followed by a specific rule (make cleanup). In this case, only the rule *cleanup* is called (I normally call this rule *clean*). *Note: echo is a command to output to screen*

makefile

```
executable: main.o mouse.o   #TARGET, PREREQs (notice here the prereqs are targets made
below)
        @g++ -std=c++11 -o main1 main.o mouse.o #COMMAND

main.o: main.cpp #TARGET, PREREQs
        @g++ -std=c++11 -c main.cpp #COMMAND

mouse.o: mouse.cpp #TARGET, PREREQs
        @g++ -std=c++11 -c mouse.cpp #COMMAND

cleanup:
        echo "Cleaning up object files and executable!" #output to screen
        rm *.o
        rm main1
```

```
computer$ make
Computers-MacBook-Air:mouse computer$ ls
a.out      main.cpp   main.o          main1      makefile
     mouse.cpp  mouse.h         mouse.o
computer$ make cleanup
echo "Cleaning up object files and executable!" #output to
screen
Cleaning up object files and executable!
rm *.o
rm main1
computer$ ls
a.out      main.cpp   makefile   mouse.cpp  mouse.h
```

As you can probably tell, the beauty of a makefile is that when we are compiling/running our program we may have multiple steps. We can put everything we want into our makefiles and just call make! The examples given above were simple, but you can imagine a more complex program can have a pretty complex makefile! You can google "c sample makefile" to see examples (and of course you can do makefiles in other languages, not just C!)

Note: If you get the following error, make sure that your *makefile* does not end with .txt AND that you are trying to compile from the same directory with your *makefile*:

```
computer$ make
make: *** No targets specified and no makefile found.  Stop.
```