$\frac{65}{100}$

You have a formatting error that prevents me from running the program

See back page and correct for a regrade

```c
#include <dirent.h>
#include <sys/stat.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>


//node struct to be used in the queue
typedef struct _node_t
{
        char *direct;
        struct _node_t *next;
        struct _node_t *previous;
} node_t;

//global variables for the queue
node_t *head = NULL;
node_t *tail = NULL;
node_t *current;
int queuesize = 0;

// int isempty(){          //used to tell if the queue is empty
//     if(head ==NULL){
//         return 0;
//     }
//     return 1;
// }

void enqueue(char *directory){ //sets the directory name at the end of the queue
    node_t *current=malloc(sizeof(node_t));
    int length = strlen(directory)+1;
    current->direct = malloc(length);
    strncpy(current->direct, directory, length);

    if (queuesize == 0){
        tail = current;
        head = current;
    }else{
        tail->next = current;
        current->previous = tail;
        tail = current;
    }
    queuesize++;

}


char *dequeue(){     // returns the directory name stored in the head and delete the head node pushing head back a node in the queue

    current = head;
    char *toreturn=malloc(512);
    strncpy(toreturn, head->direct, 512);


    if(queuesize>1){
        head = current->next;
        current->next = NULL;
        head->previous = NULL;
        current->previous=NULL;
    }else{
        head = NULL;
        tail = NULL;
        current->next = NULL;
        current->previous = NULL;
    }

    queuesize--;
    //free up all memory from the deleted node
      free(current->direct);
      free(current->next);
      free(current->previous);
      free(current);

    return toreturn;

}

int main(int argc, char **argv) {
chdir(argv[1]);//get the current working directory which we will add the relative file to
    char cwd[4096];// buffer for current working directory
    getcwd(cwd, sizeof(cwd));
    char *direc = malloc(512);

    //adding the relative file
    strcat(cwd, argv[1]);
    strcat(cwd, "/");

    DIR *directory = opendir(cwd);
    struct dirent *entry;
     enqueue(cwd);

    while(queuesize >0){
        memset(direc, 0, 512); //reset direc
        printf("\n%s\n", head->direct); //print the directory name
        direc = dequeue();  //get the directory to open

        if (chdir(direc) == -1){ //this will give an error if the file could not be read and print reason
            perror("Error: ");
    }else{


        if ((directory = opendir(direc)) == NULL) //if file cant open this will print the reason
        {
            perror("Cannot open .");

        }else{

            while ((entry = readdir(directory)) != NULL) //read until there are no more file or folders in the directory
            {

                if (entry->d_type == DT_DIR)    //if type is a directory set it up in the format of a path and enqueue it
                {
                    memset(cwd, 0, 4096);
                    getcwd(cwd, sizeof(cwd));
                    printf("    Directory: %s\n", entry->d_name);
                    int result = strcmp(".", entry->d_name);
                    //don't want to try to go into "." or ".."
                    if(result == 0){continue;}
```

```
                     result = strcmp("..", entry->d_name);
                    if (result == 0)
                    {
                        continue;
                    }
                    strcat(cwd, "/");
                    strcat(cwd, entry->d_name);
                    strcat(cwd, "/");
                    enqueue(cwd);
                    memset(cwd, '\0', 512);
                    continue;
                };
                if (entry->d_type == DT_REG) // if type is folder just print name
                {
                    printf("    File: %s\n", entry->d_name);
                    continue;
                };
        }
        closedir(directory);
            }
        }
        printf("---------------------------\n"); // after each file just print a line
        }

        return 0;
}
```

```
test ran on testdir

Error: : No such file or directory

/home/git/clones/3240/a2/corey.s.oldenberg/testdir./testdir/
---------------------------

test ran on etc copy

Error: : No such file or directory

/home/git/clones/3240/a2/corey.s.oldenberg/etc./etc/
---------------------------

valgrind results for testdir

==8243== Memcheck, a memory error detector
==8243== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==8243== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==8243== Command: ./a.out ./testdir
==8243==
Error: : No such file or directory
==8243==
==8243== HEAP SUMMARY:
==8243==     in use at exit: 1,024 bytes in 2 blocks
==8243==   total heap usage: 5 allocs, 3 frees, 5,205 bytes allocated
==8243==
==8243== LEAK SUMMARY:
==8243==    definitely lost: 1,024 bytes in 2 blocks
==8243==    indirectly lost: 0 bytes in 0 blocks
==8243==      possibly lost: 0 bytes in 0 blocks
==8243==    still reachable: 0 bytes in 0 blocks
==8243==         suppressed: 0 bytes in 0 blocks
==8243== Rerun with --leak-check=full to see details of leaked memory
==8243==
==8243== For counts of detected and suppressed errors, rerun with: -v
==8243== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

valgrind results for etc

==8245== Memcheck, a memory error detector
==8245== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==8245== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==8245== Command: ./a.out ./etc
==8245==
Error: : No such file or directory
==8245==
==8245== HEAP SUMMARY:
==8245==     in use at exit: 1,024 bytes in 2 blocks
==8245==   total heap usage: 5 allocs, 3 frees, 5,197 bytes allocated
==8245==
==8245== LEAK SUMMARY:
==8245==    definitely lost: 1,024 bytes in 2 blocks
==8245==    indirectly lost: 0 bytes in 0 blocks
==8245==      possibly lost: 0 bytes in 0 blocks
==8245==    still reachable: 0 bytes in 0 blocks
==8245==         suppressed: 0 bytes in 0 blocks
==8245== Rerun with --leak-check=full to see details of leaked memory
==8245==
==8245== For counts of detected and suppressed errors, rerun with: -v
==8245== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```