Question 1. [10 MARKS]

Part (a) [7 MARKS]

```
struct award {
    char *name;
    char **nominees;
};

/* Complete the function below that will allocate just enough memory on the heap
 * for nominee to store name, and will set nominee to name
 */
void set_nominee(char **nominee, char *name) {
    *nominee = malloc(strlen(name) + 1);
    strncpy(*nominee, name, strlen(name) + 1);
}

int main() {

    char line[1024];
    // Declare a struct award variable
    struct award bp;

    // Initialize the award's name to the read-only string "Best Picture"
    bp.name = "Best picture";

    // Initialize the award's nominees to refer to heap-allocated space for
    // 4 nominees
    bp.nominees = malloc(4 * sizeof(char *));


    printf("Enter a film name:\n");
    fgets(line, 1024, stdin); // assume line is long enough and fgets succeeds

    // Use set_nominee to initialize nominee 0 to the string stored in line
    set_nominee(&bp.nominees[0], line);

    // Use set_nominee to initialize nominee 1 to the read-only string "Moonlight"
    set_nominee(&bp.nominees[2], "Moonlight");

    printf("Award: %s\n", bp.name);
    printf("Nominees:\n");
    for(int i = 0; i < 4; i++) {
        if(bp.nominees != NULL ) {
            printf("   %s\n", bp.nominees[i]);
        }
    }
```

Part (b) [3 MARKS]

Check the box beside each of the free statements below that correctly free memory used in the above program. For each line that is not checked, briefly explan why the statement is incorrect.

☑ `free(bp.name);`

☑ `free(bp.nominees[0]);`

☑ `free(bp.nominees[1]);`

☐ `free(bp.nominees[2]);`
Nothing was allocated here

☑ `free(bp.nominees);`

☐ `free(bp);`
bp is a struct and was not dynamically allocated

Question 2. [3 MARKS]

Given the following Makefile

```
labs.gf : lab1.gf lab2.gf
    gather lab1.gf lab2.gf > labs.gf

lab1.gf : lab1.csv classlist
    mkgrades lab1.csv < classlist

lab2.gf : lab2.csv classlist
    mkgrades lab2.csv < classlist

clean :
    rm *.gf
```

The contents of the current working directory:

```
    lab1.csv    lab2.csv    classlist
```

Specfiy which files are created, deleted or modified when the following commands are run in sequence.

The question did not explain what mkgrades modified or created. When we marked it we looked for a consistent interpretation across the 3 subquestions.

Possible interpretations:

- mkgrades modified no files in which case nothing is created because lab1.gf and lab2.gf never exist.
- mkgrades creates the corresponing .gf file for the .csv file passed in. This was the intention and the answer is given below.
- mkgrades also modifies the .csv file.

It is not reasonable to assume that classlist is ever modified, and none of the rules delete a file.

Part (a) [1 MARK] `make lab2.gf`

Created	Modified	Deleted
lab2.gf		

Part (b) [1 MARK] `make`

Created	Modified	Deleted
lab1.gf labs.gf		

Part (c) [1 MARK] Suppose `lab1.csv` is modified, and then `make` is run again with no arguments. Which files are created modified or deleted?

Created	Modified	Deleted
	lab1.gf lab1.csv labs.gf	

Question 3. [1 MARK]

When I run

```
gcc -Wall -g -o prog prog.c
```

I get the error message

```
/u/reid/tmp/prog.c:5: undefined reference to ‘foo’
```

Exaplain what I need to do to address this problem.

Add to the compile line the file that contains the function or variable foo.
Or, add the reference for the function or variable foo to prog.c
Or, add the appropriate include file to prog.c

Question 4. [2 MARKS]

I spent a long time working on my `print_ftree` program on cdf and it works perfectly, but when I try it on my Linux machine at home, once it a while it gives me a segmentation fault.

Part (a) [1 MARK] Is there a bug in my code? Explain your answer. *Yes! The bug may not be noticed because the memory that was being used incorrectly coincidently had the correct values or was coincidentally not used in the first case.*

Part (b) [1 MARK] When does the segmentation fault message appear? Check the appropriate answer or answers.

- ☐ Compile time
- ☑ Run time
- ☐ When runnning `make print_ftree`

Question 5. [4 MARKS]

For each of the code fragments below, there is missing code. At the very least, the line (or lines) that declare and possibly initialize the variable `x` are missing. If the code will not compile no matter what you put for the missing code, check `COMPILE ERROR` and explain why. If the code will compile, but is not guaranteed to run without an error, check `RUN-TIME ERROR` and explain why. Otherwise, check `NO ERROR` and give the correct declaration for x. You don't need to show any other missing code. The first one is done for you.

Code Fragment	ERROR	Declaration for `x` or explanation

`int y;`
`// missing code`
`x = y;`
	☑ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		`int x;`

`char **n = malloc(3*sizeof(char *));`
`// missing code`
`strcpy(n[0], x);`
	☐ NO ERROR
	☐ COMPILE ERROR
	☑ RUN-TIME ERROR
		No memory has been declard
		for n[0] to point to

`char course[7];`
`x = "csc209";`
`// missing code`
`course = x;`
	☐ NO ERROR
	☑ COMPILE ERROR
	☐ RUN-TIME ERROR
		Can't change the value of the array course

`char *film = "La La Land";`
`// missing code`
`char *x = *film + 6;`
	☐ NO ERROR
	☑ COMPILE ERROR
	☐ RUN-TIME ERROR
		Note that char + int is fine.
		but type mismatch in assignment of
		int to char *

`// struct definition used`
`// for the last two snippets`
`struct Node {`
`  char *fname;`
`  struct Node *next`
`};`		

`x = malloc(sizeof(struct Node));`
`// missing code`
`x.fname =  "file1.txt";`
	☐ NO ERROR
	☑ COMPILE ERROR
	☐ RUN-TIME ERROR
		Can't use . notation with pointers
		(or no declaration for x)

`struct Node *root;`
`root = malloc(sizeof(Node));`
`// missing code`
`x = &root->next;`
	☐ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		(Compile error due to sroot or sizeof(Node))
		struct Node **x;
		Operator precedence means
		`&root->next == &(root->next)`

Question 6. [6 MARKS]

Complete the function below, so that the following examples will work correctly and the minimum amount of memory is used. The only string library functions you may use are `strlen` and `strncpy`. (You are not required to use these functions.)

```
char *path1 = "/";
char *result1 = dirname(path1);

char *path2 = "/usr/include";
char *result2 = dirname(path2);

char path3[9] = "file.txt";
char *result3 = dirname(path3);

char path4[20] = "a2/test/test1.in";
char *result4 = dirname(path4);

printf("%s, %s, %s, %s\n",
     result1, result2, result3, result4);
```

Prints:

```
/, /usr, ., a2/test
```

```
/* Returns the component(s) of path up to but not including the final '/'
 * path will not be NULL
 * If there is no '/' then dirname returns the string "."
 */
char *dirname(char *path) {

     char *result;
     int i = strlen(path) - 1;
     while(i > 0 && path[i] != '/') {
         i--;
     }
     if(i == 0)  {
         if(path[i] == '/') {
             // could malloc here
             return path;
         } else {  // no slash so return current directory
             result = malloc(2);
             strncpy(result, ".", 2);
             return result;
         }
     }
     result = malloc(i+1);
     for(int j = 0; j <= i; j++) {  // could use strncpy here
         result[j] = path[j];
     }
     result[j-1] = '\0';
     return result;
 }
```


END OF SOLUTIONS