Question 1. [5 MARKS]

Assume you have a terminal open, and the current working directory contains a C program file called `wizard.c` and a text file called `potion.txt`.

Part (a) [1 MARK]

The following command is used to compile `wizard.c` into an executable:
`gcc -Wall -o spell -g -std=99 wizard.c`

Write a shell command to execute the executable file produced by the statement above with command line argument `confundo` and write the output to `potter` instead of to standard out.

```
./spell confundo > potter
OR
spell confundo > potter
```

Part (b) [1 MARK]

Give a one-line shell command to change the permissions on `spell` so that is readable by everyone, executable by the users who owns the file and members of the group and writeable by no-one.

```
chmod 554 spell
```

Part (c) [3 MARKS]

You have a binary file containing only 100 consecutive integers such that the first integer is at the beginning of the file (starting in byte 0.) Add to the code fragment below, to efficiently read the value for the 42nd integer into the variable `i`;

```
int i;
FILE *fp = fopen("someBinaryFile", "r");

fseek(fp, 41 * sizeof(int), SEEK_SET);
      fread(&i, sizeof(int), 1, fp);

OR
fseek(fp, 41 * sizeof(int), SEEK_CUR);
      fread(&i, sizeof(int), 1, fp);
```

Question 2. [6 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;`

`int **p = malloc(3 * sizeof(int *));`
`// missing code`
`x = *(p + 1);`
	☑ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		`int *x;`

`struct person {`
`  char *name;`
`};`
`// missing code`
`x->name = "???";`
	☑ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		`struct person *x`
		`[= malloc(sizeof(struct person*));]`

`int **p = malloc(sizeof(int**));`
`// missing code`
`*p = &x;`
	☑ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		`int x;`

`char *s = malloc(6 * sizeof(char));`
`strcpy(s, "hello");`
`// missing code`
`x = **s;`
	☐ NO ERROR
	☑ COMPILE ERROR
	☐ RUN-TIME ERROR
		`Cannot dereference char * twice.`

`char *course = "csc207";`
`// missing code`
`x = course;`
`x[5] = '9';`
	☐ NO ERROR
	☐ COMPILE ERROR
	☑ RUN-TIME ERROR
		`Cannot change a string literal.`

`void fun(XXX type_hidden) {`
`  // missing code`
`  x[1] = type_hidden[2];`
`}`
`int main() {`
`  int array[3] = {1, 7, -4};`
`  fun(array);`
	☑ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		`int *x [= malloc(sizeof(int) * 3);]`
		`OR x[2];`

Question 3. [8 MARKS]

Complete this program that uses a string in the format `"MonthName DayNumber"` (e.g. `December 25`), creates an `event` struct with the fields set based on the string, and prints the information in a different format.

```
struct event {
    char *month;
    int day;
};

/* Set the month and day of an event struct given a string in the format
 * "MonthName DayNumber"
 *
 * For example, if arg is "December 25", the event's month should be set to "December",
 * and its day should be set to 25.
 */
void set_struct(struct event *e, char *arg) {
    char *tmp = strchr(arg, ' ');
    e->date = strtol(tmp + 1, NULL, 10);

    int size = strlen(arg) - strlen(tmp) + 1;
    e->month = malloc(sizeof(char) * size);
    strncpy(e->month, arg, size - 1);
    e->month[size] = '\0';
}

/* Return the integer representation of month. For example, given the string
 * "January", get_month_as_num should return 1.  If month is not a valid month,
 * (not in months below), return 0.
 */
int get_month_as_num(char *month) {
    char *months[] = {"January", "February", "March", "April", "May", "June",
        "July", "August", "September", "October", "November", "December"};

    for (int i = 0; i < 12; i++) {
        if (strcmp(month, months[i]) == 0) {
            return i + 1;
        }
    }
}
```

```
int main(int argc, char **argv) {
    struct event *new_event = malloc(sizeof(struct event));

    // Call set_struct() on new_event:
    set_struct(new_event, argv[1]);

    // Print the event information in the following format:
    //     Month: December
    //     Date: 25
    //     Converted: 12/25
    // You will want to use get_month_as_num() when printing the last line.

    printf("Month: %s\nDate: %d\nConverted: %d/%d\n",
        new_event->month, new_event->date, get_month_as_num(new_event->month),
        new_event->date);

    // Free all dynamically-allocated memory:
    free(new_event->month);
    free(new_event);

    return 0;
}
```

Question 4. [6 MARKS]

The function `intertwine()` takes two strings as arguments, and returns a dynamically allocated string containing a mix of both arguments. Characters in the returned string should be added in the following order: first character of the first argument, first character of the second argument, second character of the first argument, second character of the second argument, etc. until all characters have been added.

Some examples of input and output can be seen below:

```
intertwine("hello", "world") -> "hweolrllod"
intertwine("go on" "odmring") -> "good morning"
intertwine("CSC209", "") -> "CSC209"
```

You may assume that both `s1` and `s2` are strings. That is, they are null-terminated.

Write the function `intertwine()` according to the above description.

```
char *intertwine(char *s1, char *s2) {

    int size_s1 = strlen(s1);
    int size_s2 = strlen(s2);
    int size_ret = size_s1 + size_s2;
    char *retval = malloc(sizeof(char) * (size_ret + 1));

    int i = 0;
    int idx = 0;
    while (i < size_s1 || i < size_s2) {
        if (i < size_s1) {
            retval[idx] = s1[i];
            idx++;
        }
        if (i < size_s2) {
            retval[idx] = s2[i];
            idx++;
        }
        i++;
    }

    return retval;
}
```

ALTERNATE SOLUTION:

```
    retval[size_ret] = '\0';

    return retval;
    char *result = malloc(strlen(s1) + strlen(s2) + 1);
    int min = strlen(s1);
    if (strlen(s2) < min) {
        min = strlen(s2);
    }

    for (int i = 0; i < min; i++) {
        result[i*2] = s1[i];
        result[i*2 + 1] = s2[i];
    }

    result[min*2] = '\0';
    strcat(result, &s1[min]);
    strcat(result, &s2[min]);

    return result;
```