Question 1. [4 MARKS]

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

Part (a) [2 MARKS]

Write a shell command that compiles `compile_me.c` into an executable called `compiled`. Include the flag to display all warning messages.

```
gcc -Wall -o compiled compile_me.c
```

Part (b) [2 MARKS]

Write a shell command to run the executable produced in part (a) and redirect its standard output to a file called `my_output.txt` in the *parent* directory of the current working directory.

```
./compiled > ../my_output.txt
```

Question 2. [4 MARKS]

For each code fragment below, the declaration of the variable `x` is missing.

If the code will not compile no matter what you put for the missing code, check `COMPILE ERROR` and explain why. If it is possible to write a declaration to make the code compile without errors, but running the program *might* cause an error, check `RUN-TIME ERROR` and explain why. Otherwise, check `NO ERROR` and write the correct declaration for `x`. The first one is done for you.

Note: the only missing code is the declaration of `x`. All other code is shown.

Code Fragment	ERROR	Declaration for `x` or explanation

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

`int **p = malloc(3 * sizeof(int *));`
`// missing declaration`
`x = **(p + 1);`
	☐ NO ERROR
	☐ COMPILE ERROR
	☑ RUN-TIME ERROR
		`*(p + 1)` isn't initialized, so dereferencing it may lead to an error

`char word[4];`
`// missing declaration`
`strcpy(x, "hey");`
`word = x;`
	☐ NO ERROR
	☑ COMPILE ERROR
	☐ RUN-TIME ERROR
		`word` is type `char[4]` and cannot be reassigned

`char *greeting = "hello";`
`// missing declaration`
`x = greeting;`
`x[0] = 'j';`
	☐ NO ERROR
	☐ COMPILE ERROR
	☑ RUN-TIME ERROR
		Cannot change a string literal.

`int *p = malloc(2 * sizeof(int));`
`int m = 1;`
`p = &m;`
`// missing declaration`
`*(x + 1) = p;`
	☑ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		`int *x[2];`

Question 3. [4 MARKS]

Write function `sample_string` that takes a string `s` and an integer `k` and returns a dynamically-allocated string that is built by concatenating every k^{th} character from `s` starting from position 0. You should not allocate more space for the result than is necessary.

For example, `sample_string("abcdefg", 2)` should return the string "aceg".

You may assume that `k >= 1`.

```
char *sample_string(char *s, int k) {
```

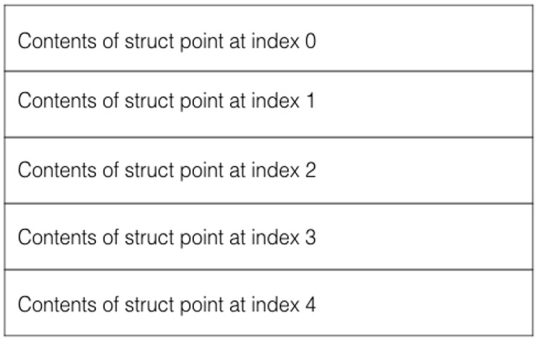
There are other possible solutions. This one has been checked.

```
    int space_needed = strlen(s)/k;
    if(strlen(s) % k != 0) {
        space_needed += 1;
    }

    printf("Space needed is %d\n", space_needed);
    char *result = malloc(space_needed + 1);
    for (int i=0; i<strlen(s); i+=k) {
        result[i/k] = s[i];
    }
    result[space_needed] = '\0';
    return result;
```

Question 4. [6 MARKS]

The program below takes the file name of a binary file as an argment. The binary file contains an array of structs with the type shown below. Assume that the format is correct and there are enough elements in the array. An example of the format is shown to the right.
Complete the program that will read the struct stored at index 3, add 10 to the `x` field of the struct, and write the struct back to the file in exactly the same location. Do not read the whole array of structs into memory.




```
struct point {
    char label[24];
    int x;
    int y;
};

int main(int argc, char **arg) {

    FILE *fp = fopen(argv[1], "r+");  // but don't penalize for the mode

    fseek(fp, sizeof(struct point) * 3, SEEK_SET);
    struct point p;
    fread(&p, sizeof(struct point), 1, fp);
    p.x += 10;
    fseek(fp, sizeof(struct point) * 3, SEEK_SET);
    fwrite(&p, sizeof(struct point), 1, fp);

    return 0;
```

Question 5. [5 MARKS]

Write the `repeat` function below according to its description.

```
typedef struct node {
    char *item;
    struct node *next;
} Node;

/*
 * Given the head of a linked list where each node stores a string,
 * modify the list so each node has its string repeated n times
 * where n is assumed to be >= 1.
 *
 * For example, if the list originally contains the strings
 * {"david", "karen", "csc209"}, after calling this function with n = 2,
 * the list should contain the strings
 * {"daviddavid", "karenkaren", "csc209csc209"}.
 *
 * You will need to use malloc to allocate space for each new string.
 * Carefully calculate how much space is needed for each new string.
 *
 * You do *not* need to free anything here.
 */
void repeat(Node *head, int n) {

    Node *curr = head;
    while (curr != NULL) {
        char *new = malloc(n * strlen(curr->item) + 1);
        new[0] = '\0';
        for (int i = 0; i < n; i++) {
            strcat(new, curr->item);
        }
        curr->item = new;
        cur = cur->next;
    }

}
```