Question 1. [4 MARKS]

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

Part (a) [2 MARKS]

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

```
gcc -Wall -o doit calculate.c   (optionally could include -std flag as well)
```

Part (b) [2 MARKS]

Given the following makefile rule, explain what would cause the action of the rule to be executed when we run `make testout`. Assume that the files `testout`, `doit`, and `grades.txt` all exist in the current working directory.

```
testout : doit grades.txt
    doit grades.text > testout
```

*A*t least one of doit or grades.txt must be newer than testout.

Question 2. [4 MARKS]

For each code fragment below, if the code will not compile or will generate a warning when compiled with the `-Wall` flag, 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 show what is printed. The first one is done for you.

Code Fragment	ERROR	Output or explanation for error

`int y = 2;`
`int x = y;`
`printf("%d %d", x, y);`
	☑NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		`2 2`

`char word[4];`
`char *thing = "box";`
`word = thing;`
`printf("%s", word);`
	☐ NO ERROR
	☑COMPILE ERROR
	☐ RUN-TIME ERROR
		`incompatible types`
		`(char[4] vs. char *)`
		`Cannot assign to array name`
		`only array element`

`char **w = malloc(2 * sizeof(char *));`
`w[0] = "hello";`
`*(w + 1) = w[0] + 1;`
`printf("%s, %s", w[0], w[1]);`
	☑NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		`hello, ello`

`char name[] = "You";`
`char *person = name;`
`printf("%s\n", person);`
`free(person);`
	☐ NO ERROR
	☐ COMPILE ERROR
	☑RUN-TIME ERROR
		person points to name, which exists on stack and cannot be freed
		Cannot free memory that was not dynamically allocated.

`char *names[2];`
`names[0] = malloc(5 * sizeof(char));`
`strncpy(names[0], "Jane", 5);`
`names[1] = "John";`
`for (int i = 0; i < 5; i++) {`
`  names[0][i] = 'A';`
`  names[1][i] = 'B';`
`}`
`printf("%s\n%s", names[0], names[1]);`
	☐ NO ERROR
	☐ COMPILE ERROR
	☑RUN-TIME ERROR
		names[1] cannot be mutated because it points to a string literal

Question 3. [4 MARKS]

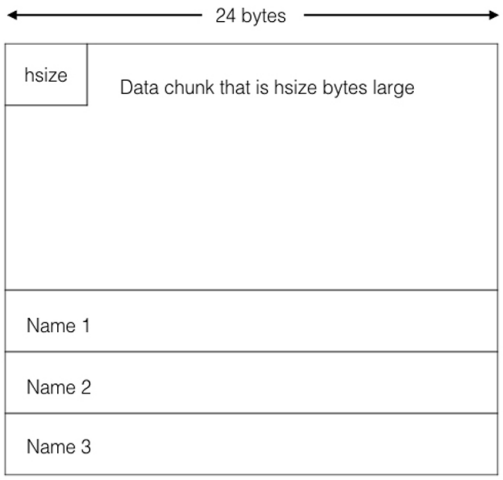
Write the `terminate_string()` function below according to its description. Notice that variable `c` is not necessarily a string before the function is called.

```
/* From the index of the first null terminator in character array c up to and
 * including index idx, replace each character with a null terminator.
 *
 * If c does not contain a null terminator within the characters from indexes
 * 0 to idx inclusive, replaces the charcter at index idx with a null terminator.
 *
 * Return the index of the first null terminator in c after the
 * replacement(s).
 *
 * Assume that idx >= 0.
 */
int terminate_string(char *c, int idx) {

    int saw_null = 0;
    for (int x = 0; x < idx; x++) {
        if (saw_null) {
            str[x] = '\0';
        } else {
            if (str[x] == '\0') {
                saw_null = x;
            }
        }
    }

    str[idx] = '\0';
    if(saw_null == 0) {
        return idx;
    } else {
        return saw_null;
    }
}
```

Question 4. [6 MARKS]



You are given a binary file with the following format:

- The first four bytes of the file are an integer that contains the size of the data chunk that follows, not including the first 4 bytes.
- After the data chunk is a sequence of strings representing names. Each string is stored in a 24 byte character array.

The layout of the file is shown in the diagram on the right.

Complete the program below that takes a file name of the above format as a command line argument, and prints to standard output the second string in the sequence.

```
int main(int argc, char **argv) {


    FILE *fp = fopen(argv[1], "r");

    int hsize;
    fread(&hsize, sizeof(int), 1, fp);
    //printf("hsize = %d\n", headersize);

    fseek(fp, headersize, SEEK_CUR);
    fseek(fp, 24, SEEK_CUR);
    char name[24];
    fread(name, 1, 24, fp);
    printf("%s\n", name);
    return 0;
```

Question 5. [5 MARKS]

Write the `concat` 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,
 * return a dynamically-allocated string obtained by concatenating
 * each string in the linked list, in order they appear in the list.
 *
 * You may assume the linked list is non-empty, and contains only
 * valid strings.
 *
 * Do this in two passes through the linked list:
 *   1. First, calculate the total amount of space needed to store
 *      the output string.
 *   2. Allocate space for the string, build the final result, and return.
 */
char *concat(Node *head) {

    Node *curr = head;
    int size = 0;
    while (curr != NULL) {
        size += strlen(curr->item);
        curr = curr->next;
    }

    char *result = malloc(size + 1);
    result[0] = '\0';
    curr = head;
    while (curr != NULL) {
        strcat(result, curr->item);
    }

    return result;
}
```