Question 1. [8 MARKS]

Complete the code below according to the comments. The struct represents a person in a family.

```
struct person {
    char *name;
    struct person *children[100]; // Every person has at most 100 children.
};

int main() {
    // Declare a struct person variable (stack-allocated).
    struct person p;


    // Initialize the person's name to the read-only string "Eliza", and each
    child pointer to NULL.
    p.name = "Eliza";
    for (int i = 0; i < 100; i++) {
        p.children[i] = NULL;
    }

    // Set the first three child pointers to refer to different
    // heap-allocated blocks of memory. Each block should be exactly the
    // size required to store a struct person.
    // Do not initialize these three children.
    p.children[0] = malloc(sizeof(struct person));
    p.children[1] = malloc(sizeof(struct person));
    p.children[2] = malloc(sizeof(struct person));

    // Initialize the first child's name to a heap-allocated string "Will".
    p.children[0]->name = malloc(5);
    strcpy(p.children[0]->name, "Will");

    // Free all dynamically-allocated memory you used in the parts above.
    free(p.children[0]->name);
    free(p.children[0]);
    free(p.children[1]);
    free(p.children[2]);

    return 0; }
```

Question 2. [3 marks]

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

Part (a) [1 mark]

Write a command to compile `vote.c` into an executable called `vote`, including debugging symbols and using the c99 standard.

```
gcc -Wall -o vote -g -std=c99 vote.c
```

Part (b) [2 marks]

Run a command to run `vote` with command line arguments `david` and `michelle`, and redirect the contents of file `input.txt` to `vote`'s standard input.

```
vote david michelle < input.txt
OR
./vote michelle david < input.txt
```

Question 3. [3 marks]

For each of the code fragments below, the goal is for the variable `sisters` to hold the value 3. For each independent fragment, indicate if:

- It would not compile - explain why.
- It would compile but might have a warning and might have a run-time error - explain why.
- It would compile and run but sisters would not be 3 - say what the value in sisters *would* be.
- The code compiles and runs and sisters hold value 3 - no additional info required in this case.

Code Fragment	Result	Additional Info

`int sisters;`
`int *sis_pt = &sisters;`
`sisters = 3;`
`*sis_pt = 6;`
	□ would not compile
	□ warning or possible run-time error
	☑ runs but sisters is not 3
	□ runs and sisters holds value 3
		value of sisters is 6

`int sisters = 1;`
`int *sis_pt;`
`sis_pt= &sisters;`
`*sis_pt = 3;`
	□ would not compile
	□ warning or possible run-time error
	□ runs but sisters is not 3
	☑ runs and sisters holds value 3	

`int sisters;`
`int *sis_pt;`
`*sis_pt = 3;`
`sisters = *sis_pt;`
	□ would not compile
	☑ warning or possible run-time error
	□ runs but sisters is not 3
	□ runs and sisters holds value 3
		can't dereference sis_pt before initializing it

Question 4. [4 marks]

For the program below, each time a variable is declared or memory is otherwise allocated, write the amount of memory that is allocated, where it is allocated, and when the memory is de-allocated. For stack memory, specify which stack frame the memory belongs to. Note: some programs allocate more than one block of memory.

Code Fragment	Amount of memory	Where?	De-allocated when?
`int *fun(int x) {`	1 integer	stack - fun	end of fun
`  int *p = malloc(sizeof(int)*2);`	1 integer pointer	stack - fun	end of fun
`  *p = x;`	2 integers	heap	end of main
`  return p;`			
`}`			
`int main() {`			
`  int *s = fun(10);`	integer pointer	stack - main	end of main
`  return 0;`			
`}`			

Question 5. [7 marks]

The zipper of string `s1` and `s2` is defined to be a string created by concatenating alternating characters from each string (starting from `s1` until one string reaches the end and then appending the remaining chars from the other. Here are some examples:

s1	s2	zipper of s1 and s2
“ABCD”	“abcd”	“AaBbCcDd”
“ABCD”	“ab”	“AaBbCD”
“AB”	“abcd”	“AaBbcd”

On the next page, write the function `zip` according to its documentation. Notice that `s1` and `s2` are `const` - your function must not mutate them.

```
/*
 Return the pointer to a dynamically allocated string that is the zipper of s1 and s2.
 Precondition: s1 and s2 are both null-terminated strings
 */
char * zip(const char *s1, const char *s2) {


    // This is only one solution. There are lots of ways to solve this problem.

    char * result = malloc(strlen(s1) + strlen(s2) + 1);
    int i;
    for (i = 0; i < shorter_len(s1, s2); i++) {
        result[2*i] = s1[i];
        result[2*i + 1] = s2[i];
    }

    // make result a string so we can use strcat
    result[2*i] = '\0';

    // now tack on left over
    // at least one of s1[i] or s2[i] == '\0'
    if (s1[i] == '\0') {
        strcat(result, &s2[i]);
    } else {
        strcat(result, &s1[i]);
    }
    return result;
}

// Using a helper function is not required
int shorter_len(const char * s1, const char * s2) {
    if (strlen(s1) > strlen(s2)) {
        return strlen(s2);
    } else {
        return strlen(s1);
    }
}
```