Question 1. [8 marks]

Complete the code below according to the comments. Store **all** strings on the heap.

```
// This struct represents information about a game player.

struct player
    char *name;
    int rank;
    char *recently_played[3]; // Names of people this player has recently played
;

int main()
    // Allocate space on the heap for a struct player, and save the pointer
    // to this memory in a variable.
    struct player *p;
    p = malloc(sizeof(struct player));


    // Initialize the player's name to "Bobby", rank to 1,
    p->name = malloc(6);
    strcpy(p->name, "Bobby");
    p->rank = 1;

    // and one recently-played name to "Garry".
    p->recently_played[0] = malloc(6);
    strcpy(p->recently_played[0], "Garry");


    // Free the space allocated for the struct, and any other memory
    // which was dynamically-allocated.
    free(p->name);
    free(p->recently_played[0]);
    free(p);

    return 0;
```

Question 2. [3 MARKS]

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

Part (a) [1 MARK]

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

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

Part (b) [2 MARKS]

Write a command to execute `foo` with command line arguments `david` and `michelle`, and redirect the output to the file `output.txt`.

```
foo david michelle > output.txt
OR
./foo michelle david > output.txt
```

Question 3. [5 MARKS]

Part (a) [2 MARKS]

In the boxes below, write the value of the expressions `depth`, `x`, and `*p3` at the points in the program execution indicated by the boxes' positions in the code.

```
int depth = 4;
int *p3 = &depth;
int x;
x = *p3 + 5;
```

depth	x	*p3
4	9	4

```
depth = 1;
```

depth	x	*p3
1	9	1

Part (b) [3 MARKS]

Show the output of each printf statement in the corresponding box.

```
char c[6] = "ABCDE";
char *p = c;
char *s = p + 2;
```

`printf("%c\n", p[0]);`	A
`printf("%s\n", p + 1);`	BCDE
`printf("%c\n", s[0]);`	C

Question 4. [6 marks]

Part (a) [4 marks]

Complete the following function below according to its documentation.
Suggestion: Use `strstr` - the man page excerpt is in the API.

```
/*
 Replace the first complete occurence of string piece in string s1 with X's.
 Return 0 if successful and 1 if piece does not occur in s1.
 Precondition: piece and s1 are both null-terminated strings.

   Example: if s1 is "hello" and piece is "el", s1 is changed to become "hXXlo".
 */
int find_and_mask(char *s1, const char *piece) {

    char *loc = strstr(s1, piece);
    if (loc == NULL) {
        return 1;
    } else {
        // replace starting at loc for length of piece with X's
        for (int i = 0; i < strlen(piece); i++) {
            *loc = 'X';
            loc++;
        }
        return 0;
    }
}
```

For each of the following calls to `find_and_mask`, indicate whether the call is correct, will definitely cause an error, or may cause an error. Either explain the error, or show the output if it is correct.

Part (b) [1 mark]

```
char outer[8] = "password";
char *hidden_bit = "ss";
find_and_mask(outer, hidden_bit);
printf("%s\n", outer);
```

Circle **one**: Runs Will cause error [✓ May cause error] `outer` is not null-terminated

Part (c) [1 mark]

```
char *full = "Hide my secret please.";
char *bit = "secret";
printf("%s\n", full);
find_and_mask(full, bit);
printf("%s\n", full);
```

Circle **one**: Runs [✓ Will cause error] May cause error `full` is read only

Question 5. [3 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. Include memory allocated for string literals.

Code Fragment	Amount of memory	Where?	De-allocated when?
`int main() {`			
`  char *name = "David";`	size of(char *)	stack - main	end of main/program
	6 * size of(char)	read only memory	end of main/program
`  char *c[5];`	5 x sizeof(char *)	stack - main	end of main/program
`  c[2] = name;`			
`  c[3] = malloc(12);`	12 bytes	heap	end of main/program
`  return 0;`			
`}`			