1 Programming in C and C++ (AVSM)

A spacecraft arrives at Mars, but its memory has been corrupted by radiation en route. Luckily, it can receive updates one bit at a time using a predefined C function

```c
short receive_bit(void)
```

that when called will return either 1 or 0. The stream of bits for a value is transmitted in unsigned big-endian byte order: for example, a 16-bit value of 125 would be `0000000000111101`. Assume the `int` type is 32 bits.

(a) Explain the meaning of the `inline` keyword on C function declarations, and a potential drawback of using it. [2 marks]

(b) Using `receive_bit()`, define a function `receive_int()` that decodes and returns a 32-bit value from the sequence of received bits. [4 marks]

(c) Build a more general decoding function `receive` using a C++ template with two parameters that specify the number of bits to decode and a datatype for the decoded value. Use this to write two template instantiations that decode an 8-bit value into a `short` and a 32-bit value into an `unsigned long`. [6 marks]

(d) Find and explain four instances of undefined behaviour that could result from compiling and running the C code below with different command-line arguments. The `strcpy(dst,src)` function copies a zero-terminated C string from the `src` buffer to the `dst` buffer. The `putchar(c)` function outputs a character `c` to the console. You can assume that the standard C header prototypes have been included for `<stdio.h>`, `<stdlib.h>` and `<string.h>`. [8 marks]

```c
1. char *show_instruction(int msg) {
2. char buf[6];
3. int fuel;
4. if (msg == 1 && fuel--) {
5. strcpy(buf, "THRUST");
6. return buf;
7. } else if (msg == 2) {
8. char *msg = (char*)malloc(100);
9. strcpy(msg, "DEPLOY_PARACHUTE");
10. return msg;
11. }
12. }
13. 
14. int main(int argc, char **argv) {
15. char *msg;
16. msg = show_instruction(argc);
17. putchar(msg[0]);
18. return 0;
19. }
```