1. Declare a function prototype for a function named `my_func1` that accepts an integer as input parameter and returns an integer.

Answer:
```
int my_func1(int i);
// i can be replaced by any valid identifier

Or:
int my_func1(int);
```

2. Declare a function prototype for a function named `my_func2` that accepts 2 integers as input parameters and returns a float.

Answer:
```
float my_func2(int i, int j);
// i and j can be replaced by any valid identifier

Or:
float my_func2(int, int);
```

3. Implement a function with prototype `int is_even(unsigned int num)` that returns 0 if `num` is odd, or 1 if `num` is even.

Answer:
```
int is_even(int num)
{
      if(num%2 == 0) return 1;
      return 0;
}
```

4. Consider the following C function definition:

```
int sum(int a, int b)
{
   return a + b;
}
```

Rewrite the function as a function-like macro.

Answer:
```
#define sum(A, B)     ((A)+(B))
```

5. Declare the following:

   a) An array of characters named `achar` which can hold 10 characters.
   b) An array of characters named `bchar` which can hold 10 characters, with the first 3 characters initialized to `'A'`, `'B'`, and `'C'`, respectively, and the rest initialized to the null character.
   c) An array of integers named `cint` which can hold 5 integers.
   d) An array of integers named `dint` which can hold 5 integers, with all the values initialized to 0.
   e) A two-dimensional array of long integers named `elong` with 4 rows and 5 columns.
   f) A symbolic string literal named `WARNING` with value `"Enter at your own risk."` using macro.
   g) A symbolic string literal named `ERROR` with value `"Incorrect."` using const.
   h) A string variable named `player_name`, which can hold 32 characters and initialized to `"Bob"`.

Answers:

```
a) char achar[10];
b) char bchar[10] = {'A', 'B', 'C'};
c) int cint[5];
d) int dint[5] = {};
e) long elong[4][5];
f) #define WARNING "Enter at your own risk."
g) const char *ERROR = "Incorrect.";
h) char player_name[32] = "Bob";
```

6. Consider the following C statement:

```
char str[10] = "Hello";
```

   a) What is the size (in bytes) of the character array `str`?
   b) How many characters are occupied by the string `"Hello"`?
   c) What is the length of the string `"Hello"`?
   d) What is the index of the character `'o'`?
   e) Write an assignment statement to replace the character `'H' with 'y';`

Answers:

a) 10 bytes
b) 6 (includes the null character itself)
c) 5 (excludes the null character)
d) 4
e) `str[0] = 'y';`

7. Determine whether the following are valid or invalid string literals:

```
a) "Hello, world"
b) "Hello,\tworld"
c) 'Hello, word'
d) 'H'
e) "Hello" ", " "world"
f) "Hello, \
world"
```

Answers:
a) Valid
b) Valid
c) Invalid
d) Invalid
e) Valid
f) Valid

8. Consider the following declarations:

```
int c = 'Y';
char message[20] = "Welcome";
```

Write a C statement using `printf()` to

a) Print `c` as character.
b) Print `c` in decimal form.
c) Print `c` in octal form.
d) Print `c` in hexadecimal form.
e) Print the `message` string.

Answers:

```
a) printf("%c", c);
b) printf("%d", c);
c) printf("%o", c);
d) printf("%x", c);
e) printf("%s", message);
```

9. Consider the following declarations:

```
int c;
char message[20];
```

Write a C statement using `scanf()` to

a) Read input as character and store in variable `c`.
b) Read input as decimal and store in variable `c`.
c) Read input as octal and store in variable `c`.
d) Read input as hexadecimal and store in variable `c`.
e) Read input as string and store in variable `message`.

Answers:

```
a) scanf("%c", &c);
b) scanf("%d", &c);
c) scanf("%o", &c);
d) scanf("%x", &c);
e) scanf("%s", message);
```