1. Given the following declaration, write a snippet of C code that might lead to `strlen(arr)` returning no less than 8.

```c
char arr[4];
```

2. Fill in the correct expression:

```c
char s1[MAX1];
char s2[MAX2];
getname(s2, MAX2); /* Initializes the string s2 */
strncpy(s1, s2, ____________________);
```

3. a) Fill in the argument for malloc so that it allocates just enough space for the remaining code.

```c
char **s = malloc(_____________________________);
char p[10] = "Paul";
char q[10] = "Karen";
char r[10] = "Francois";
*s = p;
*(s+1) = q;
*(s+2) = r;
```

b) Write the above 3 statements using array notation so that they have the same effect.

c) Write one C statement to truncate the string "Francois" so that the following printf statement prints Fran

```c
printf("\%s\n", r);
```

d) Give the type of the following expressions. If the expression is not a pointer, also give its value.
4. Given the two declarations below circle the C statements that will compile without warning or error (other than those about unused variables):

```c
int *p;
int i = 10;
char q = i;  char *c = p;  double *f = &i;  double d = i;
```

5. Show what is written to the file for each of the `fprintf` and `fwrite` statements. Show the value(s) in decimal and binary. ASCII values for characters: '0' is 48 (0x30), '1' is 49 (0x31), '6' is 54 (0x36)

```c
int i = 16;
fprintf(fp, "%d", i);

int j = 0x10;
fprintf(fp, "%d", j);

fwrite(&i, sizeof(int), 1, fp);
fwrite(&c, sizeof(char), 1, fp);
```