1. What will happen if, in a C program, you assign a value to an array element whose index exceeds the size of array (e.g. A[15] = 8 while A is an array of 14 elements only)? You must specify what will be the compiler reaction, and, if the compilation is successful, what will happen during execution.

2. In C, if you pass an array as an argument to a function, what actually gets passed?

3. What will be the output of the following program? Explain how you reached your solution.

```c
#include<stdio.h>

int main()
{
    int a[5] = {5, 1, 15, 20, 25};
    int x, y, z;
    x = ++a[1];
    y = a[1]++;
    z = a[x++];
    printf("%d, %d, %d", x, y, z);
    return 0;
}
```

4. We have seen in class that a[i] is equivalent to *(a+i). What is the equivalent of a[i][j]?

5. The function `strstr()` finds the first occurrence of a substring in another string. It is declared as: `char *strstr(const char *s1, const char *s2);` On success, strstr returns a pointer to the element in s1 where s2 begins (points to s2 in s1). If s2 does not occur in s1, strstr returns null.
   a) Without using strstr, implement this function in C. Optimize it as much as you can.
   b) Design a test to see whether your implementation is faster/slower than `strstr`. Clearly explain your test, how you conducted it, and what your findings are.

6. Is there anything wrong with the following declaration? If so, what is it?

```c
struct emp
{
    int ecode;
    struct emp *e;
};
```
7. Is there anything wrong with the following code? If so, what is it?

```c
struct emp
{
    int ecode;
    struct emp e;
};
```

8. Assume we are executing the following program on a 32-bit machine:

   a) What will be the output?

   b) If we change p’s declaration to be: `double * p;` and use the type casting with `malloc` to be `(double *)`, what will be the output?

```c
#include<stdio.h>
#include<stdlib.h>

int main()
{
    int *p;
    p = (int *)malloc(20);
    printf("%d\n", sizeof(p));
    free(p);
    return 0;
}
```