History and Evolution of Programming Languages

1. Explain the relationship between machine language, assembly language, and high level languages.

Differences in Language Paradigms

2. (a) List a difference between imperative/procedural programming and object-oriented programming.

   (b) List a difference that functional programming has from procedural or object-oriented programming.

Programming In C

3. Explain the purpose of each of the following GCC options:

   - `-o`
   - `-c`
   - `-std`
   - `-Wall`

Modular Design and Development

4. For each of the following snippets of code, state whether it should be placed in the header file (`.h`) or the source file (`.c`).

(a)
```
struct Point
{
    double x;
    double y;
};
```

(b)
```
int main(int argc, char **argv)
{
    int x = run_some_function();
    printf("%d\n", x);
    return 0;
}
```

(c)
```
int do_something_and_return(int x, int y)
{
    if (y != 0)
        return x*y + (x/y);
    else
        return x*y;
}
```

(d)
```
int do_something_and_return(int x, int y);
void do_something(int x, int y);
```

5. Give an example of a header guard for a header file named `linkedlist.h`.

Variables and Basic Data Types

6. List at least 7 of the basic data types in C (not including the unsigned variants).

Initialization and Coercion

7. There are at least two issues in the following code. What are they?

```
#include <stdio.h>

int main(void) {
    int numbers[] = { 3, 6, 4, 5, 14, 9 };
    int sum, i = 0;
    double average;
    for (; i < 6; ++i) {
        sum += numbers[i];
    }
    average = sum / 6;
    printf("%f", average);
    return 0;
}
```

Strings

8. (a) If you want to store the string ``Hello, world!'' in the `str` variable in the following code, what is the minimum acceptable value of `n`? Why?

```
int n;
...
char str[n];
```

   (b) What is the character literal for the null terminator?

Arrays

9. (a) What is the issue with the code below? What is the result of the issue?

```
 1 int i, x;
 2 int arr[10];
 3 for (i=0; i < 10; ++i)
 4 {
 5     scanf("%d", &x); // Reads a number from stdin and stores it in x
 6     arr[i] = x;
 7 }
 8
 9 for (i=10; i >= 0; --i)
10 {
11     printf("%d\n", arr[i]);
12 }
```

   (b) What is the intended behavior of the program?

I/O

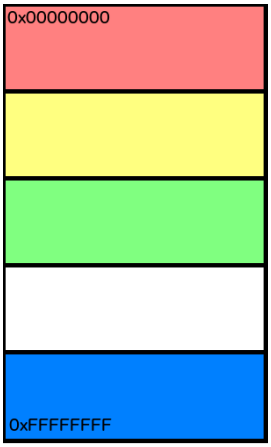
10. There is a problem with the following code as written that will manifest itself for certain user input. Explain what this issue is and how to solve it.

```
#include <stdio.h>

int main(void) {
        const int BUF_SIZE = 256;
        char buf[BUF_SIZE];
        while (fgets(buf, BUF_SIZE, stdin)) {
                printf(buf);
        }
}
```

Memory and Program Layout

11. Label the sections of the program in memory and describe what is stored in each. Your options are: `stack, data, heap, unallocated space, text`.



C Pre-processor

12. Given the following, what is output of this program?

```
#define THING1 40
#define THING2 32
#define THING4 i

#if THING1 < THING2
#define THING3 5
#elif THING2 < THING1
#define THING3 6
#else
#define THING3 7
#endif

int main() {
    int THING4 = THING3;
    printf("%d", i);
}
```

C Pointers

13. (a) What does the following function do?

```
1  int foo(int n, int *arr, int **bestp) {
2      int *start;
3      int *end;
4      int best = 0;
5      *bestp = arr;
6
7      for (start = arr; start < arr + n; ++start) {
8          for (end = start; end < arr + n && *end == *start; ++end);
9          if ((end - start) > best) {
10             best = (end - start);
11             *bestp = start;
12         }
13         start = end - 1;
14     }
15     return best;
16 }
```

(b) Make a memory map of `foo`. Use the first value set to each variable in the map.

(c) Given the following code in `main`, write code that calls our `foo` function from above and prints the result.

```
int main(int argc, char **argv)
{
    int n, i;
    int arr[] = {1, 1, 1, 2, 2, 2, 5, 5, 5, 5};

    // Write your code here.









    return 0;
}
```

Structs, Dynamic Storage

14. The following program compiles.

(a) Will the program crash at run time? If so, on which line will it crash?

```
 1 #include <stdlib.h>
 2 int main(int argc, char **argv)
 3 {
 4     int *x = NULL;
 5     int *y = NULL;
 6     int *z = NULL;
 7
 8     x = (int *) malloc(sizeof(int) * 10);
 9     y = (int *) malloc(20);
10     x = (int *) malloc(sizeof(char) * 50);
11
12     free(x);
13     free(y);
14     free(z);
15
16     return 0;
17 }
```

(b) What tool can you use to find memory leaks? What options would you use?

(c) What output would you get from part b? Is there a memory leak? (If so, where?)