C Preprocessor

1. Consider the following program:

```
1  #include <stdio.h>
2
3  #ifdef WINDOWS
4  #include <windows.h>
5
6  #define WIN_CREATEFILE(a) CreateFile(a, \
7                                       GENERIC_READ, \
8                                       FILE_SHARE_READ, \
9                                       OPEN_ALWAYS, \
10                                      FILE_ATTRIBUTE_NORMAL)
11 #endif
12
13 int main(int argc, char** argv) {
14     // ... some previous initialization stuff
15
16 #ifdef WINDOWS
17     HANDLE file = WIN_CREATEFILE(argv[1]);
18 #else
19     FILE* file = fopen(argv[1], "r");
20 #endif
21
22     // ... do some more stuff
23
24     return 0;
25 }
```

   (a) What flag would you use to enable the Windows specific build?

   (b) Why is this useful?

   (c) Why is the macro useful?

Memory Management

2. Consider the following statement in a larger program:

```
int* x = (int*) malloc(20); //create an array of 20 ints
```

   After testing the program, you notice that the values of $x[5], x[6], \cdots, x[19]$ keep changing unexpectedly.

   (a) Why is this?
   (b) What should the statement actually be?

3. 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?)

4. (a) Given the following:

```
struct Point {
    char label;
    double x;
    double y;
};
```

What (specifically) happens when the following command is executed?

```
struct Point *newPoint = (struct Point *) malloc( sizeof(struct Point) );
```

(b) Let's add some more information:

```
typedef struct {
    struct Point p1;
    struct Point p2;
    struct Point p3;
} Triangle;
```

What (specifically) happens when the following command is executed?

```
Triangle *tri = (Triangle *) malloc( sizeof(Triangle) );
```

Program Translation

5. (a) List the four main steps in the *program* translation process.

   i.

   ii.

   iii.

   iv.

   (b) List the four phases of *source code* translation, which make up the compilation step of program translation.

   i.

   ii.

   iii.

   iv.

Abstract Data Types

6. Generally speaking, ADTs are easier to define and work with in procedural languages like C, as opposed to object-oriented languages like Java or C#. (**True** or **False**). Explain your answer.

Makefiles

7. Consider the following makefile:

```
1  CFLAGS := -std=c99 -Wall -Wextra
2  me: me.o
3      $(CC) $(LDFLAGS) -o $@ $^ $(LDLIBS)
4  calc: calc.o real.o
5      $(CC) $(LDFLAGS) -o $@ $^ $(LDLIBS)
6  calc.o: calc.c
7      $(CC) $(CFLAGS) -c $<
8  real.o: real.c
9      $(CC) $(CFLAGS) -c $<
10 .PHONY: clean
11 clean:
12     $(RM) me calc *.o
```

   (a) Ralph is annoying. One day, when Ralph makes a particularly unreasonable demand, his friend loses it and shouts, "Why don't you make me???" Always one to take things literally, Ralph pops open his favorite Bourne-compatible shell and types: `make me`. However, he is confronted with the message: `make: 'me' is up to date`.

   Explain the meaning of this message. Which (if any) of the relevant files are now located in Ralph's directory? What numeric values did `Make` compare before outputting this message?

   (b) Ralph is childishly proud of himself, but everyone just groans and tells him to "get real". Coincidentally, Ralph has a library providing functions for working with reals, as well as a calculator program to test the functionality. Ready to win another trivial victory, Ralph types `make clean` and then `make calc`.

   List the exact sequence of commands that are executed as a result of this new invocation.

File I/O

8. Write a program that searches a file for a provided number on `stdin`. Print out any errors on `stderr`. Example:

```
$ fileSearch file.txt
> 234
found: 234
```

9. (a) The following program is intended to read a text file and outputs (as binary data) the number of characters (including new lines) in each line to a file. However, there is a problem with the code the way it is currently written. Find the problem and explain how to fix it. Note: The code is compiled and linked using GCC.

```
 1 #define _GNU_SOURCE
 2
 3 #include <stdio.h>
 4 #include <stdlib.h>
 5
 6 int main(void) {
 7     FILE* inputfile = fopen("input.txt", "r");
 8     if(!inputfile) {
 9         perror("fopen failed for inputfile:");
10         exit(EXIT_FAILURE);
11     }
12     int numchars = 0;
13     char* line = NULL;
14     FILE* outputfile = fopen("output.txt", "w");
15     if(!outputfile) {
16         perror("fopen failed for outputfile:");
17         fclose(inputfile);
18         exit(EXIT_FAILURE);
19     }
20     while((numchars = getline(line, 1024, inputfile)) != -1) {
21         fprintf(outputfile, "%d\n", numchars);
22     }
23     fclose(inputfile);
24     fclose(outputfile);
25     return EXIT_SUCCESS;
26 }
```

(b) Rewrite the code so that it outputs the data using binary streams instead.