Q1

A common method for compiling a C program is:

```
gcc -Wall –std=c99 file.c
```

This creates a file named a.out. What does the -Wall flag do?

1. It checks to make sure the entire program is correct.
2. It checks for logic errors.
3. It reports back with all warning messages.
4. It creates an object file.

Q3

A segmentation fault occurs when:

1. a compiler finds a syntax error.
2. a program uses a recursive function that fills the stack.
3. a program attempts to access a memory location it is not allowed to.
4. a program doesn't use functions.

Q4

Consider the following code and compiler error message:

```
7  int x
8  x = 2;

main.c:8: error: syntax error before 'x'
```

What is wrong with this code?

1. Line 8 should be **x := 2;**
2. Line 8 should be **x=2;**
3. Line 7 is missing a semicolon to terminate it.
4. There is no syntax error.

Q5

What is missing from the usability of the following printf statement?

```
printf("Size of air conditioner: ");
scanf("%lf", &ac_TONS);
```

1. An indicator of what size is (litres).
2. An indicator of the volume (length x width x height).
3. An indicator of what size is (TONS).
4. Nothing, the usability is just fine.

Q6

What is the most appropriate datatype for the following variable?

```
Number of rain barrels: numRainB
```

1. double
2. int
3. char
4. long long

Q7

What is the purpose of defensive programming? (>1 answer):

1. To check user input for invalid data.
2. To ensure a piece of software continues to function despite unforeseeable usage.
3. To reduce syntax errors.
4. To stop the user from using the program properly.

Q8

To print the following string as a prompt using a `printf` statement:

```
Interest rate (%):
```

What is used to print the character **%** ?

1. `\%`
2. `^%`
3. `%%`
4. `%`

Q12

What is the purpose of functions? (>1 answer)

1. Functions can be collected into a library.
2. Functions make it easier to document the program.
3. Functions parallelize the programs processing.
4. Functions implement the idea of program modularity.

Q13

What is the purpose of the `.2` in the following statement?

```
printf("The tree contains %.2fkg of carbon\n", treeC);
```

1. It prints the value stored in `treeC` in a column two characters wide.
2. It prints the double value stored in `treeC` to 2 decimal places.
3. It converts the double value stored in `treeC` to an integer.
4. It multiplies the value of the variable by 0.2 before it prints it out.

Q14

How many times will the following loop execute?

```
int i, n=1000;
for (i=1; i<n; i=i+1){
    frac = numer / denom;
    x = x + frac;
    numer = -numer;
    denom = denom + 2.0;
    i = i - 1;
}
```

1. 1000
2. 999
3. To infinity and beyond
4. 0

Q15

Consider the following code segment:

```
int x;
char c;
scanf("%d", &x);
printf("%d", x);
scanf("%c", &c);
printf("%c", c);
```

When the code is compiled and executed, the following output is obtained (when an attempt is made to enter the number 12 and the character z).

```
>./a.out
12
12

>
```

Q15

The program fails to allow the user to enter a value for `z`, and terminates. What is wrong with the program?

1. The buffer is flushed.
2. The <return> (entered after typing 12) is stored in the buffer and incorrectly read into variable c.
3. A stack overflow causes the buffer to be purged.
4. It was coded using the language of moisture evaporators.

Q16

Consider the following function definition:

```
void treeCarbon(double bioA, double bioR, double *TC)
{
        *TC = 0.5 * (bioA + bioR);
}
```

What is the appropriate function call given that the variables `bA`, `bR`, and `carbon` are all doubles:

1. `treeCarbon (bA, bR, *carbon);`
2. `carbon = treeCarbon (bA, bR);`
3. `carbon = treeCarbon (bA, bR, carbon);`
4. `treeCarbon (bA, bR, &carbon);`

Q17

What sort of error is contained in the following defensive code?

```
while (1){
    printf("%% of the roof emptying into barrels (1-100%)? ");
    scanf("%lf", &roofP);
    if (roofP >= 1 || roofP <= 100)
        break;
}
```

1. A logic error.
2. A syntax error.
3. A fandango on core.
4. A stack breach.

Q19

Consider the following code which reads in a string:

```
char wood[9];
printf("softwood or hardwood? ");
scanf("%s", wood);
```

A problem with `scanf` is that a user can type beyond the allowable 8 characters of the string wood, and it will be stored (somewhere in memory).
What is a better way of inputing a string?

1. `fscanf(“%s”, wood);`
2. `wood = getchar();`
3. `scanf(“%c”, wood);`
4. `fgets(wood, 9, stdin);`

Q20

Consider the following function header:

```
int palindrome(char a[100], int n);
```

Choose the appropriate function call given the following information in the calling function:

```
char p[100];
int N=100, isP;
```

1. `isP = palindrome(N, p);`
2. `palindrome(p[100], N);`
3. `isP = palindrome(p, &N);`
4. `isP = palindrome(p, N);`

Q21

Consider the following piece of code:

```
int main(void)
{
    main();
    return 0;
}
```

This code is an example of what?

1. a linked list
2. a simulated stack-inflicted function call
3. recursion
4. a regressive function

Q22

Continue to consider the following piece of code:

```
int main(void)
{
    main();
    return 0;
}
```

What will happen when this code is run?

1. infinite recursion
2. a segmentation fault
3. a stack overflow
4. both 2. and 3.

Q23

The keyword `stdin` stands for what?

1. standard pointer
2. strict din number
3. standardized dining
4. standard input

Q24

Consider the following piece of code:

```
char str[100];
fgets(str,100,stdin);
```

If the following string is input, what is stored in `str`:

```
winter is coming, run!
```

1. “winter\n”
2. “winteriscoming!\n\0”
3. “winter is coming, run!\n\0”
4. “winter is coming!\0”

Q25

What does this program fragment do?

```
s = 0;
for (i=n; i >= 1; i = i - 1)
    if (n % 2 == 0)
        s = s + i;
```

1. It computes the sum of the integers from 1 through n.
2. It computes the sum of the integers from 1 through n-1.
3. It computes the sum of the even integers from 1 through n.
4. It computes the sum of the odd integers from 1 through n.
5. Maybe, kinda, sorta, none of the above?

Q1

The following preprocessor directive is correct:

```
#define LAMBDA = 2.99792e8
```

True or false?

Q36

Array indexing in C starts at:

1. 1
2. 0
3. an arbitrary value declared when the array is created.
4. the starting index depends on the architecture of the microprocessor.

Q34

Which loop checks the test condition at the end of the loop?

1. for
2. while
3. do-while
4. repeat-until
5. no looping process checks the test condition at the end

Q35

The `rand()` built-in library function:

1. is a pseudo-random number generator.
2. returns positive double values.
3. is a true random number generator.
4. works together with a lava lamp to produce random numbers.

Q33

The library function `toupper()` requires the header file:

1. `<string.h>`
2. `<stdlib.h>`
3. `<ctype.h>`
4. `<time.h>`

Q25

Random numbers generated by computers are not really random, they are considered pseudo-random.

True or false?

Q31

A recursive function is a function that:

1. returns itself (inside out).
2. takes itself as a parameter.
3. calls itself.
4. is parasitic and lives inside other functions.

Q30

Which one of the following is not a benefit of functions?

1. Modularization.
2. To allow certain information to be hidden.
3. Avoiding code repetition.
4. Making a program more efficient (speed-wise).

Q29

Of the following statements, which one initializes all the 10 elements in an int array named rainfall to zero:

```
1.  int rainfall[10];
2.  rainfall[*] = 0;
3.  for (i=0; i<10; i=i+1)
       rainfall[i] = 0;
4.  rainfall[1:10] = 0;
```

Q28

Which of the following statements should you use to copy the contents of the string text into the string text2:

```
char text[50];
char text2[100];
```

1. `strcpy(text2, text);`
2. `strcpy(text, text2);`
3. `text2 = text;`
4. `*text2 = strcpy(text);`

Q27

Suppose we have the following string declaration:

```
char str[5];
```

Which string below can be correctly assigned to the variable **str**?

1. `yavin`
2. `alderran`
3. `hoth`
4. `tatooine`

Q26

In a group of nested loops, which loop is executed the most number of times?

1. the outermost loop
2. the innermost loop
3. all loops are executed the same number of times
4. cannot be determined without knowing the size of the loops

Q3

C is case sensitive, meaning that the variables `PI`, `Pi`, and `pi` are all different.

True or false?

Q21

Numerical calculations using floating-point numbers are more precise than those using integers.

True or false?

Q20

The following is a valid array subscript:

```
double b[1000];
b[1000] = 53.1;
```

True or false?

Q19

If a program stack is filled to capacity, and overflows, then the program will likely crash.

True or false?

Q18

The function, **atoi** is used to convert a string into an integer.

True or false?

Q11

What are the basic datatypes in C:

1. `int, single, double, char`
2. `integer, real, boolean, char`
3. `int, float, double, char`
4. `long, long long, double double`

Q7

The correct way of calculating 2^n is, and assigning its value to `x` is:

```
1. x = 2 * n;
2. x = 2 ^ n;
3. x = pow(2,n);
4. x = 2n;
```

Q2

The expression **23/4** performs integer division.

True or false?

Q39

Once the following string has been declared

```
char DNA[20];
```

which one of the following statements is a valid initialization?

1. `DNA = “ATGCTGA”;`
2. `DNA[1:7] = “ATGCTGA”;`
3. `strcpy(DNA,”ATGCTGA”);`
4. none are valid.

Q5

```
int m;
double e=9.0;
```

In the statement

```
m = e + 3.14159;
```

`m` is equal to 12.

True or false?

Q37

What will be printed when the code below is executed?

```
int x = 0;
for (x=1; x<7; x=x+2);
    printf("x=%d\n", x);
```

1. x=0
2. x=1, x=3, x=5, x=7
3. x=3, x=5
4. x=7

Q38

When calling a function, an array should be passed using this form:

1. the name of the array followed by square brackets.
2. the name of the array preceded by an asterisk.
3. the name of the array alone.
4. the name of the array followed by square brackets containing i.

Q48

What is the main reason to avoid using global variables?

1. They cause spaghetti code.
2. They cause syntax errors.
3. Their non-locality: they can potentially be modified from anywhere.
4. No reason, they are good programming practice.

Q43

Consider the following code:

```
char str[15];
scanf("%s", str);
printf("%s", str);
```

What is printed out if we enter:

```
The quick brown fox runs over the lazy dog.
```

1. `The quick brown`
2. `The`
3. `Thequickbrownfo`
4. `The quick brown fox runs over the lazy dog.`

Q44

What would be stored in b if we run the following code:

```
int a[5]={1,2,3,4,5};
int b;
b = a;
```

1. nothing - this is not a valid C expression
2. `b` will contain a copy of the values in `a`
3. `b` will contain point to the same memory location as `a`

Q45

What does the following function do?

```
void WHAT(int *a, int *b)
{
    *a = *a + *b;
    *b = *a - *b;
    *a = *a - *b;
}
```

1. It calculates the Greatest Common Divisor of a and b.
2. It sorts the numbers a and b.
3. It swaps two values without using a temporary variable.
4. None of the above.

Q49

If a program contains the following code:

```
int min=24, sec=53;
double d_minutes;
d_minutes = min + sec/60;
```

What is the value of `d_minutes`:

1. 24
2. 24.0
3. 24.9
4. 24.883333333333333333333333

Q50

This function is an example of:

1. a Schwartzian transform.
2. recursion.
3. iteration.
4. trial and error.

```
void hailstone(int x)
{
    printf("%d ", x);
    if (x != 1)
        if (isodd(x))
            hailstone(3*x+1);
        else
            hailstone(x/2);
}
```

Q51-53

```
1   #include <stdio.h>
2   #include <ctype.h>
3
4   int main(void)
5   {
6       int i;
7       char phoneN[14];
8       printf("Enter phone number> ");
9       scanf("%13s", ________);
10
11      for (i=0; i<13; i=i+1)
12          if (_______(phoneN[i]) || phoneN[i] == '-')
13              continue;
14          else if (isalpha(phoneN[i]))
15              convAlpha2Dgt(________);
16          else {
17              printf("Error - invalid character in input\n");
18              return 0;
19             }
20      printf("The numeric phone number is %s\n", phoneN);
21      return 0;
22  }
```

The function **convAlpha2Dgt** converts an alphabetic character (A-Z) to the corresponding digit (2-9) using phone letter-number mapping - it takes a character as input, and the parameter is pass-by-reference. e.g. the program input is **1-800-THRIFTY** and the program output is **1-800-8474389**.

Q51

What goes in the blank part of the statement on line 9 of the program?

1. `phoneN[]`
2. `*phoneN`
3. `phoneN`
4. None of the above.

Q52

What goes in the blank part of the statement on line 15 of the program?

1. `phoneN[i]`
2. `&phoneN[i]`
3. `*phoneN[i]`
4. None of the above.

Q53

What goes in the blank part of the statement on line 12 of the program?

1. `isdigit`
2. `isalpha`
3. `iscntrl`
4. None of the above.

Q54-56

```
1  void read_daytime(char day[4], char hour[3], char mins[3])
2  {
3     char daytime[12];
4     printf("Enter the day time DDDHH:MM ");
5     fgets(daytime,12,stdin);
6     sscanf(daytime, "%3s%2s:%2s", day, hour, mins);
7  }
```

Q54

If the function is called and the following is input when prompted:
WED07:42:59

What is stored in day, hour and mins.

1. `day = WED, hour = 7, mins = 42`
2. `day = WED, hour = 07, mins = :42`
3. `day = WED, hour = 07, mins = 42`
4. `day = WED07, hour = 42, mins = 59`

Q55

In the main function, the following code is used to run the function.

```
char day[4], hour[3], mins[3];
read_daytime(day, hour, mins);
printf("It is %s at %d:%d\n", day, atoi(hour), atoi(mins));
```

The `atoi` function is used to convert the string into a int.

1. `true`
2. `false`
3. `maybe`

Q56

The string daytime is able to store 11 characters from the ASCII table.

1. `true`
2. `  false`
3. `maybe`