CST8234 – Midterm #1

Solutions – Parts B and C

28. A function prototype is an example of a _____forward declaration_____ _____, which indicates that the function will actually be defined later.

29. _____system_____ libraries are included with the C compiler. The _____< >_____ symbols indicate that their header files are found in the usr/include folder. By contrast, _____user-defined_____ libraries are found in the $PATH and are indicated by placing the _____" "_____ symbols around the name of the library.

30. C's assert() function is a primitive form of _____exception handler_____.

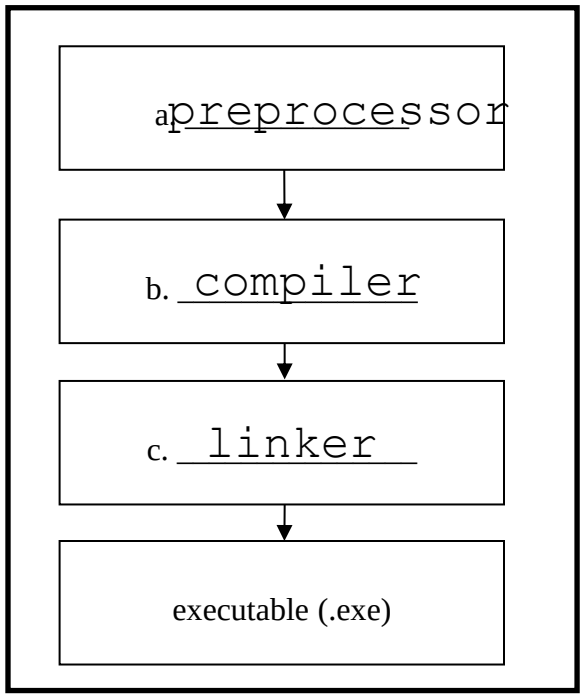
31. For a function whose definition was

```
USHRT Foo (int alice, float bob){
     //code goes here
   }
```

the corresponding function prototype would be:

USHRT Foo (int, float);

32. In the box at right indicate the three main stages of compilation that result in an executable program. Print your single-word answers in the boxes indicated. (NOTE: no acronyms or abbreviations; use complete words)




```
#include <stdio.h>
#define MAX 4

int main(void){
   int curMAX = 1, ctr;
   for (ctr = 0; ctr <= curMAX; ctr++){
      printf("*");
      if (ctr == curMAX){
         ctr = 0;
         curMAX = (curMAX++) % MAX;
         printf("\n");
      }
   }
}
```

**
***
****
*
**
***
****

34. What is the value of z after the following code executes? Explain your reasoning in the box below.

```
#include <stdio.h>
  #define FRACTION(a,b) a/b

  int main(void){
     int a = 1, b = 3, c = 2, d = 6;
     float z = 0.0;
     z = FRACTION(c+d,a+b) ;
     printf("z = %f\n", z);
  }
```

z = ____11____ because **:**
FRACTION(a,b) is a macro that defines a function that performs the operation a/b, i.e. the first argument is 'a', the second is 'b'. So
FRACTION(c+d, a+b) ->
c+d/a+b
= 2+6/1+3 = 11

35. Write a short, complete, standalone program (i.e no functions) that reads in a string and outputs each word on its own line, with the first letter of each word capitalized. You may assume the size of the string entered never exceeds 80 characters. Note: ctype.h contains the toupper() function, which takes a lower case character and converts it to upper case. (Use the last page if you run out of space.)

```
Please enter a string to capitalize, terminated
by a carriage return/enter:
capitalize this sentence
Capitalize
This
Sentence
```

```
#include <stdio.h>
#include <ctype.h>

int main(void){
   char str[80] = {'\0'}, ch;
   int chCtr = 0;
   printf ("Enter a string : ");
   scanf("%s", str);

   do {
       ch = str[chCtr];
       if (ch == ' '){
          printf("\n");
          ch = str[++chCtr];
          printf("%c",toupper(ch));
       }
       else
          printf("%c", ch);
       chCtr++;
   }while (ch);
   printf("\n");
}
```

36. Write a **single** define macro (in the box below) that replaces each of the repetitive parts of the code at right (high-lighted in bold) with a single line of code. Your macro should allow the programmer to both print out the messages indicated (i.e. the printf statements) as well as read values into the appropriate variable (the scanf statements).

NOTE: You do **not** need to show your code in use

```
#define OP(str, format, variable) \
printf("Please enter your "str"\n"); \
scanf(#format, variable);
```

```
#define OP(str, format, variable) \
printf("Please enter your "#str"\n"); \
scanf(format, variable)
```

```
char spot[] = {'\0'};
int dogsAge;
OP("dog's name", "%s", spot);
OP("dog's age", "%d", &dogsAge);}
```

37. The following code was designed to calculate the average of some number of user-entered positive integral values by first summing their total in an externally-defined function, returning that value to the main() function, and then calculating the average by dividing through by the total number of values entered by the user. The process ceases when the user enters a 0, and the average of the values entered is then output.

There are at least five compile time errors in the following code, and four logic or run-time errors (which do not show up during compilation, but cause the program to execute incorrectly). Circle any nine of these errors in the code at left, place a number from 1-9 next to the error, and describe that error in the box at right adjacent to the number you assigned in the code. Three files comprise the entire program: Main.c, Total.h, and Total.c. You can assume that the gcc compilation command is entered correctly at the command line.

Note: (1) you may assume that the fundamental construction of the code is essentially correct (sum up the values in an external function, and divide that number by the total number of values entered to get the average), even if its implementation is unnecessarily bloated and flawed in many critical areas. Your job is not to critique the entire construction of the code, but simply to find specific compilation and run-time/logic errors. (2) If you find an error that seems to occur more than once, chances are that that code is in fact correct, and the real error lies elsewhere in another location, and only once. Even if an error does occur more than once, you only get credit for the first occurrence of that error, not for repeat errors of the same kind.

(In Main.c)

```
#include <stdio.h>
#include "Total.h"

typedef UINT unsigned int;
UINT getTotal(unsigned long);

int main(void)
  UINT total, totNumsIn;
  unsigned long newNum;

  do {
    printf("Enter a positive integer, "
           "or 0 to exit");
    scanf("%lu", newNum);
    totNumsEntered++;
    if (newNum >=0)
      total = getTotal(newNum);
    else {
      printf("Error: number entered"
        " must be 0 or greater"\n);
    }
  while (newNum)
  printf("Ave. is %u\n", total/totNumsIn);
  return(0);
}
```

(In Total.h, located in the same directory as Main.c)

```
external UINT getTotal(unsigned long);
```

(In Total.c, located in the same directory as Main.c)

```
UINT getTotal(newNum){
  unsigned long total;
  return ((UINT)(total += newNum));
}
```

Compile-time errors

- `typedef` is backwards
- Missing { after main
- `newNum` needs `&` in front in `scanf()`
- Declared `totNumsIn` but using `totNumsEntered` instead
- `\n` is outside "" in `printf()`
- Missing `;` at end of `do…while`
- `UINT` `typedef`ed in `Main.c`; not visible in the other two files
- Missing `}` before `while`
- `newNum` without type in `getTotal()`
- `getTotal()` declared twice (see Total.h)
- `external` is not a valid storage class

Run-time/logic errors

- Narrowing of cast in `Total.c`
- `totNumsIn` should be initialized to 0
- `totNumsIn` incremented even when exit condition applies, so always one greater than actual value needed
- `total` in `getTotal` should be `static`
- Loop continues even if non-positive number is entered
- `total/totNumsIn` has fractional part truncated; should be cast to `float`