CS10003 Programming and Data Structures
Spring 2024 – 2025
Class Test 1

6:45pm – 7:45pm **January 30, 2025** **Maximum Marks: 40**

[*Write your answers in the question paper itself. Answer **all** questions. **All programs must be written in C.** Fill in the blanks in the following codes to make them work as described. **Do not use extra variables.** Not all blanks carry equal marks. Evaluation will depend on overall performance.*]

1. The following program reads a positive integer n from the user, and computes the smallest power of two, that is greater than or equal to n, that is, it computes $p = 2^e$ such that $p / 2 < n \leqslant p$. The program should use math library functions to do this. The program must not contain any loop. Instead it should first compute e as $\log_2 n$ truncated to an integer. It should then compute $p = 2^e$. If $p \geqslant n$, then we are done. Otherwise, the above value of p is updated to the correct value. Fill in the blanks below so that the program behaves as mentioned above.

[10]

```
#include <stdio.h>

#include ________<math.h>________

int main ( )
{
   int n, e, p;

   printf("Enter a positive integer: "); scanf("%d", &n);

   /* First compute e */

   e = __log(n) / log(2)__[or log10(n) / log10(2)]_____________________ ;

   /* Then compute 2-to-the-power-e as p */

   p = __pow(2,e)_____________________________________________________ ;

   /* Adjust p if it is smaller than n */

   if ( _________p < n_____________ ) p = __p * 2_____________________ ;

   printf("Smallest power of 2 greater than or equal to %d is %d\n", n, p);
   return 0;
}
```

Space for Rough Work

2. The following program is meant for computing the equation of the straight line passing through two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$. The equation of the line is to be specified in the form $y = mx + c$. Two special cases need to be handled. First, if P_1 and P_2 are the same point, then this problem cannot be solved. Second, a vertical line cannot be written in the above form, and is instead printed in the form $x = c$. Assume that the coordinates x_1, y_1, x_2, y_2 are integers. However, m and c computed as above need not be integers. We use the floating-point data type `double` to store these values. Fill in the blanks below so that the program works as intended.

[10]

```
#include <stdio.h>

int main ( )
{
   int x1, y1, x2, y2;
   double m, c;

   printf("First point: ");  scanf("     %d%d     ", &x1, &y1);

   printf("Second point: "); scanf("     %d%d     ", &x2, &y2);

   if ( _______x1 == x2_____ ) {

      if ( _____y1 == y2_________ ) printf("The two points are the same\n");

      else {

         c = __x1_____________________ ;

         printf("Equation of the line is x = %lf_____\n", c);
      }

   } else {

      m = ___(double)(y2 - y1) / (double)(x2 - x1)_____________________________ ;


      c = ___y1 - m * x1___[or y2 - m * x2]_____________________________ ;

      printf("Equation of the line is y = %lf_____ x + %lf_____\n", m, c);
   }
   return 0;
}
```

Space for Rough Work

3. The following program reads a positive integer n from the user, and attempts to express n as the sum of two squares, that is, as $n = a^2 + b^2$, where a and b are positive integers with $a \leqslant b$. This inequality implies that a is no larger than the truncated integer value of $\sqrt{n/2}$. Call this maximum value of a as a_{max}. The program carries out a search for a in the sequence 1, 2, 3, ..., a_{max}. For each a, an integer b is computed as the truncated value of $\sqrt{n - a^2}$. If these integers a and b satisfy $n = a^2 + b^2$, then the search loop is broken. If none of the values of a in the above range can produce a suitable b, then also the loop is broken. At the end, the program prints its decision (failure or the values a and b). Fill in the blanks below to make the program work as explained above. Write only the `main()` function. Use math library functions whenever needed.

[10]

```
int main ( )
{
   int n, a, b, amax;

   printf("Enter a positive integer: "); scanf("%d", &n);

   amax = __sqrt(n / 2)_____ ;

   /* for loop on a in the range mentioned above */

   for ( ____a = 1_____ ; __a <= amax____ ; _____++a_____ ) {

      b = __sqrt(n - a * a)_____ ;

       /* Break if a desired b is found */

      if ( __n == a * a + b * b_____ ) break;
   }

   if ( __a > amax_____ )
      printf("%d cannot be written as the sum of two squares\n", n);
   else
      printf("%d = %d^2 + %d^2\n", n, a, b);

   return 0;
}
```

Space for rough work

4. The following program reads a positive integer n from the user. It then computes and prints the number of times each digit (decimal) appears in n. For example, if n = 250168005, then the digit 0 appears three times, the digits 1, 2, 6, and 8 once each, and the digit 5 twice. The digits 3, 4, 7, and 9 do not appear in this n. Notice that leading 0 digits should not be counted. For example, if the user enters 00250168005 as n, the digit 0 will still be reported to have appeared three times in n. You do not have to do anything special to ensure this, because `scanf` will ignore the leading zero digits while storing n. Fill in the blanks in the code below to perform the task mentioned above. Write only the `main()` function. Do not use math library functions.

[10]

```
int main ()
{
   int num, n;
   int digit, count;

   printf("Enter a positive integer: "); scanf("%d", &num);

   /* Loop over digit in the range 0 ... 9 */

   for ( ___digit = 0; digit <= 9; digit++_____________________________ ) {

      n = ___num_____________________ ;          /* Store a copy of num */

      count = ___0____________________ ;      /* Initialize count */

      /* Loop so long as all the digits of n are not considered */

      while ( ___n > 0_____________________ ) {

         if ( ___n % 10 == digit_____________ ) ++count;

         n = ___n / 10_____________________ ;  /* Update n for the next iteration */

      }
      printf("Digit %d occurred %d times\n", digit, count);
   }

   return 0;
}
```

Space for rough work