CS10003: Programming and Data Structures
Spring 2026
Class Test 1, February 05, 2026

Time: 1 Hour **Full marks: 20**

SECTION: ____ ROLL NO.: _____ NAME: _____

Q1	Q2	Q3	Q4	Q5	TOTAL
3	7	3	5	2	20

INSTRUCTIONS

1. Answer ALL questions.
2. Write the answers either within the boxes provided or on the blank lines to be filled up. Any answer written elsewhere will not be evaluated. Final answers must be written by pen, answers written in pencil will not be evaluated.

Q.1. Answer the following questions as directed.

a) Which of the following are valid datatypes in C? Write ONLY the valid datatypes in the box provided. [1]

```
double
unsigned short
string
unsigned double
```

double
unsigned short

[+0.5 for each correct answer, -0.5 for each wrong answer, minimum 0]

b) Which of the following 4 strings are valid C variable names? Write ONLY the valid variables in the box provided. [1]

```
do_while
10number_sum
_5terms
Sum-of-terms
```

do_while
_5terms

[+0.5 for each correct answer, -0.5 for each wrong answer, minimum 0]

c) What will be the value of the following expressions? [1]

```
22 + 14 * 15 / 12 % 2 * 2.5
```

24.5

```
2*3.2 - 17/2/2.5 * 4/((float) 10)/4
```

6.08

[For each expression, +0.5 for correct answer, 0 for wrong answer, binary]

Q.2. What will be printed when the following programs are executed?

a) [1]

```
#include <stdio.h>
int main( ) {
    int x = -11;
    if (x = 0) printf("red");
    else if (x < 0)
    printf("blue");
    else printf("green");
    return 0;
}
```

green

[+1 for correct answer, 0 otherwise, binary]

b) Assume that the program below is executed with input 243612. [1]

```
#include <stdio.h>
int main() {
  char x = '0', y = 0;
  while (x - '0' < 5) {
      scanf("%c", &x);
      y += x - '0';
  }
  printf("%d\n", y);
  return 0;
}
```

15

[+1 for correct answer, 0 otherwise, binary]

c) [1]

```
#include <stdio.h>
int main() {
  int x = 0, y = 5, z = 10;
  while (1) {
     x++;
     if (y > z) break;
     y += 4*x; z += 2*x;
  }
  printf("x = %d, y = %d, z = %d", x, y, z);
  return 0;
}
```

x = 3, y = 17, z = 16

**[+1 if all 3 correct,
+0.5 if any 2 correct, 0 otherwise.
Ignore x=, y= z= printing]**

d) [1]

```
#include <stdio.h>
int main()
{
     char choice = 'b'; int x = 5;
     switch(choice)
     {
         case 'a': x += 5;
         case 'b': x += 10;
         case 'c': x += 20; break;
         default: x += 100; break;
         case 'e': x += 30;
     }
     printf("%d", x); return 0;
}
```

35

[+1 for correct answer, 0 otherwise, binary]

e) [1]

```
#include <stdio.h>
int main() {
  int x = 1, y = 0, z = 1, p;
  for (p = 0; p < 10; ++p) {
        y = (x=x%2) ? z : -z;
        z++; x++;
   }
   printf("%d", y);  return 0;
}
```

-10

[+1 for correct answer, 0 otherwise, binary]

f) [1]

```
#include <stdio.h>
int main() {
   int A[7] = {0, 1, 2, 3, 0, 0, 10};
   int i = 0;
   do {
      A[i+1] = A[i] + A[i+1];
      i++;
   } while (A[i] > 0 && i < 6);
   printf("%d", A[5]);
   return 0;
}
```

6

[+1 for correct answer, 0 otherwise, binary]

g) [1]

```
#include <stdio.h>
int main() {
  int i = 0, j = 0;
  while (++i < 7)  i += ++j;
  printf("%d, %d\n", i, j);
  return 0;
}
```

10, 3

[+1 if both correct, 0.5 if any one correct, 0 otherwise.]

Q.3. An integer is a perfect square if its square root is also an integer. Fill in the blanks in the C program below so that it prints all the odd perfect squares between 1 and N (including 1 and N), where N is read from the user. Do not use any mathematical functions. [1 + 1 + 1]

```
#include <stdio.h>
int main( )
{
    int i, N, square;
    scanf ("%d", &N );
    printf ("The odd perfect squares between 1 and %d are: \n", N) ;
    for (i = 1; i <= N; i++) {

        if (i % 2 == 1) {
            square = i * i;

            if (square > N) {
                /* no further odd squares can be found
                   so no need to check anything anymore */

                break;
            } else {
                /* Print odd perfect square */
                printf("%d\n ", square) ;
            }
        }
    }
    return 0;
}
```

[For each blank, +1 for correct answer, 0 otherwise, binary.
No marks for "continue" instead of "break" in 3rd blank,
Other variations in 1st and 2nd blanks are given due credit if correct]

Q.4. Given a 1-d array X, we want to counts the number of ***even-spaced inversions*** in the array. We call a pair of indices (i, j) in the array an inversion if X[i] > X[j] AND i < j (both conditions must be true). Further, the inversion is said to be even-spaced if the difference between i and j is even. Fill up the missing code below to complete the program so that it counts the number of even-spaced inversions in the array *arr*. [1 + 1 + 2 + 1]

```
#include<stdio.h>
int main(){
   int arr[] = {10,9,8,7,6,5,4,3,2,1};
   /*Stores the count of even-spaced inversions*/

   int even_inv_cnt = ____0____;
   int i, j;
   /* Iterating over all pairs of elements*/
   for(i = 0; i < 9 ;i++) {

      for(j = i + 1; j < 10; j++){
         /*Checking if even-spaced inversions*/

         if ( (arr[i] > arr[j]) && ((j - i)%2 == 0)) {
              ++even_inv_cnt;
         }
      }
    }
    printf("Number of even-spaced inversions:%d", even_inv_cnt);
    return 0;
}
```

[For first and last blank, +1 for correct answer, 0 otherwise, binary
For the second blank, 0.5 for each of "i +1" and "j < 10"
For the third blank, 2 if fully correct, 1 if one of the conditions around && is correct, 0 otherwise
Other variations in the blanks in "for" and "if" are given due credit if correct]]

Q.5. Fill up the missing lines below so that the program initializes the array ***S*** with the characters *a*, *b*, *c*, *d*, *e*, and \n, in that order, and then prints the number of characters in ***S*** not counting the \n So it should print 5 if the code is executed. However, you are not allowed to use 5 directly anywhere. [1 + 1]

```
#include <stdio.h>
int main()
{
    int i = -1; char S[10] = {'a', 'b', 'c', 'd', 'e', '\n'};
    while (S[++i] != '\n');
    printf("The no. of characters in S is %d", i);
    return 0;
}
```

[For first blank, +1 for correct answer, 0 otherwise, binary. No marks if single quotes are not given.

For 2nd blank, +1 if fully correct, +0.5 if they have not done or done wrongly the i++, but the \n check is done, 0 otherwise]

--- The End---

ROUGH WORK ONLY
(Nothing written here will be considered for evaluation)

ROUGH WORK ONLY
(Nothing written here will be considered for evaluation)

ROUGH WORK ONLY
(Nothing written here will be considered for evaluation)