CS10003: Programming and Data Structures
Spring 2026
Class Test 2, April 2, 2026

Time: 1 Hour **Full marks: 30**

SECTION: ____ ROLL NO.: _____ NAME: _____

Q1	Q2	Q3	Q4	Q5	Q6	TOTAL
6	5	5	4	7	3	30

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.
3. You should use only malloc() for any dynamic allocation needed, do not use calloc(). Also, there is no need to free any dynamically allocated memory.

Q.1. Answer the following questions as directed.

a) A is a 16-bit signed integer. The 2's complement hexadecimal representation of A is (B87F). The 2's complement representation of (-1)*A in hexadecimal format is: _____4781____ [1]

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

b) What is the minimum and maximum signed integer that can be represented using *k*-bit 2's complement representation? [1]

Minimum: -2^{k-1} Maximum: $2^{k-1}-1$

[For each of minimum and maximum, +0.5 for correct answer, 0 otherwise]

c) Define a type called MYTYPE that represents a structure with the following fields: an integer N, and a 10-element array P of pointers to integer. [1]

```
typedef struct {
        int N;
        int *P[10];
} MYTYPE;
```

[+0.5 for correct structure definition, +0.5 for correct typedef. Ok if they define a struct first and then typdef it separately]

d) Consider the following declaration. What are the values of (*p + 2) and *(p + 2)? [1]

```
int array[10] = { 20, 18, 16, 14, 12, 10 }, *p = array;
```

22, 16

[For each, +0.5 for correct answer, 0 otherwise]

e) Consider the following declaration: int *A[20]; If **A** points to the memory location x, which memory location does **A** + 3 point to? Assume that sizeof(int) = 4 and an address takes 8 bytes. [1]

x + 24

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

f) Which of the following four assignments is/are valid after the following declaration? Write only the correct options in the box provided. [1]

```
int X[10], *Y[20], *Z;
```

(A) X = Y; (B) Y[0] = X; (C) Y = Z; (D) Z = X;

B, D

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

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

a) [1]

```
#include <stdio.h>
#include <string.h>
int main() {
  char a[20] = {'p','\0','d','\0','s','\0'}, *b =
"p\nd\ns";
  printf("%d\n%s\n",strlen(b), strcat(a, b));
  return 0;
```

5
pp
d
s

[+0.5 for 5 (strlen(b) result), +0.5 for the rest (strcat(a, b) result, only if fully correct)]

b) [1]

```
#include <stdio.h>
int main() {
   double trouble = 13.0, *pt = &trouble;
   double **ppt = &pt;
   *pt = 21.32;
   **ppt = 13.13;
   printf("trouble = %.2f \n", trouble);
   return 0;
}
```

trouble = 13.13

**[+1 for correct answer, 0 otherwise, binary.
Ok if they only say 13.13]**

c) [1]

```
#include <stdio.h>
int main() {
    int  A[] = {10, 31, 22, 54, 3, 78, 19};
    int *p = &A[3], *q = &A[6], val;
    val = (*A + (q - p)*(*(p+2)/(*(q-2))));
    printf("%d", val);
    return 0;
}
```

88

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

d) [1]

```
#include <stdio.h>
void fun(char *x, int *c) {
  char *ptr = x;
  while (*ptr != '\0') ptr++;
  *c = ptr - x;
}
int main() {
   char str[30] = "This is PDS test";
   int d;
   fun(str, &d); printf("%d", d); return 0;
}
```

16

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

e) [1]

```
#include <stdio.h>
int main() {
    int i, a[5], *j;
    for (i = 0; i < 5; i++) a[i] = i*2;
    j = a;
    for (i = 0; i < 5; i++)
       printf("%d   ", (*j)++);
    return 0;
}
```

0 1 2 3 4

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

SPACE FOR ROUGH WORK ONLY. NOTHING WRITTEN HERE WILL BE GRADED.

Q.3. The following structure stores the academic information of a student, namely name (a null-terminated string), roll number (a null-terminated string), CGPA (a float), and number of credits cleared (an int).

```
struct Stud {
        char name[40];  char roll[10];
        float CGPA;   int credit;
};
```

The function `studOfTheYear()` takes two parameters, an array of structures of type `struct Stud`, and the number of structures in the array. Fill in the following blanks so that the function prints the name and roll number of the student with highest CGPA (stored in the variable `bestCG`) among only those students who have cleared the maximum number of credits (stored in the variable `maxCredit`) among all students (if there are more than one student who have cleared the maximum number of credits and have the same highest CGPA among all such students, you can print the name and roll no. of any one of them). [5]

```
void studOfTheYear(struct Stud firstYr[], int num_of_stud)
{
        int i, maxCredit = 0, top = -1;
        float bestCG = 0.0;
        for(i = 0; i < num_of_stud; i++) {
            /* Keep track of maximum credit among all students */

            if (maxCredit < firstYr[i].credit) {
                maxCredit = firstYr[i].credit;

                bestCG = firstYr[i].CGPA ;
                top = i;
            }
            /*  Keep track of highest CGPA among students with maximum credit */
            else if (maxCredit == firstYr[i].credit) {
                if (bestCG < firstYr[i].CGPA)  {          [A]
                    bestCG = firstYr[i].CGPA ;
                    top = i;
                }
            }
        }
        printf("Name: %s\tRoll: %s\n", firstYr[top].name, firstYr[top].roll);
}
```

[For each blank: +1 for correct answer, 0 otherwise, binary.
For blank marked [A], ok if they replace "bestCGPA" with "firstYr[top].CGPA"
Some other correct variations (done by very few) are also accepted]

Q.4. Let A be a $n \times n$ square matrix. We want to do the operation $A = A - A^T$ (so the result is stored in A itself and original contents of A will be overwritten). Here A^T denotes the transpose of the matrix A. Fill in the blanks in the function below such that it takes A and its number of elements n as parameters, and replaces A with $A - A^T$, without using any additional array. You are not allowed to use any extra variables other than the ones already declared. **ROWDIM** and **COLDIM** are the maximum dimensions of a 2-d matrix that the function can handle, and $n \times n$ is the actual dimension of A. [4]

```
#define ROWDIM  100
#define COLDIM  100
void matFunc ( int A[][COLDIM], int n )
{
        int i, j;
        for (i = 0; i < n; ++i) {

                for (j = 0; j <= i; ++j) {                    [A]

                        A[i][j] = A[i][j] - A[j][i];

                        A[j][i] = -A[i][j];
                }
        }
}
```

[For each blank: +1 for correct answer, 0 otherwise, binary.
For blank marked [A], the following is also fine "j=i; j<n; j++"
Ok if the parameter is passed as "int A[ROWDIM][COLDIM]]

SPACE FOR ROUGH WORK ONLY. NOTHING WRITTEN HERE WILL BE GRADED.

Q.5. The function countArray takes an integer array A and the number of elements n in A as parameters. The function returns as its return value an integer array P, allocated dynamically within the function, which first (i.e., starting from index 0) contains all negative integers in A in reverse order from the order in which they appear in A, followed by all positive integers in A in the same order as the order in which they appear in A. Any zero present in A is not added to P. The function also returns the number of integers finally stored in P through an additional parameter. For example, if A = { –2, 4, –6, 1, 3, 0, –1, 9, 5, 0} with number of elements n = 10, the function should return (as return value) an array containing { –1, –6, –2, 4, 1, 3, 9, 5} and return the number of elements in the returned array as 8. Fill in the blanks below so that the function does this. You cannot declare any new variables. [7]

```
int *countArray ( int A[], int n, int *newCnt) {
    int *P, i, k, j, tmp;

    P = (int *) malloc (n * sizeof(int));
    *newCnt = n;    k = n - 1;   j = 0;
    for (i = n - 1; i >= 0; --i) {
            if (A[i] > 0)  P[k--] = A[i];
            else if (A[i] < 0) P[j++] = A[i];
            else {
                (*newCnt)--;
                for (tmp = k; tmp < n - 1; tmp++) P[tmp] = P[tmp + 1];
                k--;
            }
    }
    return P;
}
```

[For each blank: +1 for correct answer, 0 otherwise, binary.]
Ok if "P[k] = A[i]; k- -;" written (2 statements instead of one), same for P[j++]. Some other variations in indices of P in the assignments (done by very few) are also accepted
Correct equivalent pointer notation access for arrays is also given marks]

SPACE FOR ROUGH WORK ONLY. NOTHING WRITTEN HERE WILL BE GRADED.

Q.6. A statically declared two-dimensional character array of size 50×100 is used to store a list of names (each a null-terminated string). Each name is stored in a row. Assume that at most 49 names are stored. An empty string (only the character '\0') in a row indicates the end of the list. Complete the recursive function printNames() below to print the names stored in the two-dimensional array p (passed as parameter). [3]

```
void printNames ( char p[][100] )
{
        if (p[0][0] == '\0') return;
        printf("%s\n", p[0]);

        printNames(p + 1);
}
```

[For each blank: +1 for correct answer, 0 otherwise, binary.]
For blank 1: Ok if any other equivalent thing is written for p[0][0], as long as it correctly identifies p[0][0]. Some options are *(p[0]), *p[0], *(*p) etc.
For blank 2: Ok if any other equivalent thing is written for p[0] (the starting address of the first row), as long as it correctly identifies p[0]. Some options are *p, &p[0][0] etc.

SPACE FOR ROUGH WORK ONLY. NOTHING WRITTEN HERE WILL BE GRADED.

SPACE FOR ROUGH WORK ONLY. NOTHING WRITTEN HERE WILL BE GRADED