CS10003 Programming and Data Structures, Spring 2022–2023

Class Test 2

18–May–2023 06:45pm–07:45pm Maximum marks: 30

Roll no: _____ **Name:** _____ **Section:** ____

[*Write your answers in the question paper. Answer all questions. All questions use the programming language C.*]

1. [Recursion] Let $x_0, x_1, x_2, \ldots$ be a sequence of integers in the range $0, 1, 2, \ldots, 2022$. The user supplies two integers x and y in this range. The sequence starts with $x_0 = x$. Subsequently, for all $i \geqslant 0$, we have $x_{i+1} = x_i + 1805 \pmod{2023}$. Fill in the blanks in the program below that computes the smallest $r \geqslant 0$ such that $x_r = y$. Note that $f()$ is a ***recursive*** function. The outermost call in main() is made with the first argument $x = x_0$. If the first argument of $f()$ is x_i, then the function makes a *single* recursive call (if needed) with the first argument changed to x_{i+1}. Use this recursive logic. No credit if you use any variables (global or local) or functions other than those given below, or if you change the logic (that is, the structure) of the code. **(4)**

```
#include <stdio.h>
int f ( int x, int y )
{
    if (x == y) return ____0____ ;

    else return ____1 + f( (x + 1805) % 2023, y )____ ;
}
int main ()
{
    int x, y, r;
    scanf("%d%d", &x, &y);
    r = f(x,y);
    printf("r = %d\n", r);
    /* In order to verify that f() works correctly, print the value of x_r.
       Use a formula involving x, r (and other constants). Do not print y. */

    printf("x_r = %d\n", ____(x + 1805 * r) % 2023____ );
    return 0;
}
```

2. [Arrays]

(a) Complete the following program which initializes an array `X[]` of size 100 with integers 1–100. Elements 0 to 24 of `X[]` contain the values 76–100 sequentially. Elements 25 to 49 of `X[]` contain the values 1–25 sequentially. Elements 50 to 74 of `X[]` contain 51–75 sequentially. Elements 75 to 99 of `X[]` contain 26–50 sequentially. The program then reorders the elements of the array `X[]` such that the resulting array is sorted, that is, $X[i] = i + 1$ for all i. Note that there will be two expressions in total to answer here: the first two blanks will take the same expression, and the last two blanks will take the same expression. **(3)**

```
#include <stdio.h>

int main ()
{
    int j, t, X[100];

    /* Initialization */

    for (j=0; j<100; j++) {
        if ( j < 25 )      X[j] = j + 76;
        else if ( j < 50 ) X[j] = j - 24;
        else if ( j < 75 ) X[j] = j + 1;
        else               X[j] = j - 49;
    }
```

```
    /* Reordering */

    for (j=25; j<50; j++) {
        t = X[j];

        X[j] = ____X[j+50]____ ;

        ____X[j+50]____ = t;
    }
    for (j=0; j<25; j++)  {
        t = X[j];

        X[j] = ____X[j+75]____ ;

        ____X[j+75]____ = t;
    }
}
```

(b) You are given an array $A[\,]$ with n integers, and a target sum s. Your task is to find whether there exist two indices i and j with $0 \leqslant i < j \leqslant n-1$ such that $A[i]+A[j] = s$. The following recursive function is meant for doing that. Fill in the blanks to complete the function. The function returns *yes* or *no* (in the C fashion) depending on whether or not the equality $A[i]+A[j] = s$ is satisfied for some i, j. You must use the logic specified as comments in the code. **(6)**

```
int sumfound ( int A[], int n, int s )
{
    int i;
    if (n <= 1) return 0;     /* Base case: A[] does not have two elements. */
    /* Recursively check whether A[i]+A[j] = s for 0 <= i < j <= n-2 */

    if (sumfound( _____A_____ , _____n - 1_____ , s )) return 1;
    /* Check for the possibility whether A[i]+A[n-1] = s for some i */

    for (i = _____0_____ ; i <= _____n - 2_____ ; i++)

       if ( _____A[i] + A[n-1] == s_____ ) return 1;

    return _____0_____;
}
```

3. [Strings] Answer the following parts.

(a) Fill in the blank so that the following program fragment prints `Kharagpur`. **(1)**

```
char s[] = "IITKharagpur", *p;

    p = s + 3; or p = &s[3];
printf ("%s", p);
```

(b) What will be the output of the following program fragment? Write in the space below the fragment. **(1)**

```
char s[]={'M','y','\0','p','e','t','\0'};
strcat(s,"Tom");
printf ("%s", s);
```

MyTom

(c) The following C program takes two names from the user as input, and checks if the two names are the same or not. The programmer assumes that the length of each name is at most 20. Fill in the blanks so that the C program works correctly (under the programmer's assumption). **(4)**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main( )
{
    char *name1, *name2;
    name1 = (char *)malloc(21);
    name2 = (char *)malloc(21);

    scanf("%s", _____name1_____ );              /* read the first name */

    scanf("%s", _____name2_____ );             /* read the second name */

    if ( _____strcmp(name1,name2)_____ )  /* use a string library function */
       printf("The names are different");
    else
       printf("The names are the same");
    return 0;
}
```

A Canadian zoologist runs the program. When the program waits for reading the first name, she enters

`Gulo Gulo Vancouverensis`

and hits return. What will the program do after that? Clearly mark the correct answer among the following options.

(A) The program will wait for reading the second name.
(B) The program will encounter segmentation fault and terminate.
(C) The program will print `The names are different` and terminate.
(D) The program will print `The names are the same` and terminate.

4. [Pointers]

(a) What will be printed by the following C program segment? **(3)**

```
int a = 20, b[] = { 15, 35, 45 };
int *p, *q;
p = &a; q = &b[0];
*(q+1) = 30;
printf ("%d, %d, %d", *q+2, *(q+2), *p * b[1]);
```

Write your answer here: 17, 45, 600

(b) It is required to dynamically allocate space for an integer array with 50 elements, and initialize it with $1, 2, \ldots, 50$. Fill in the blanks so that the following C program segment works correctly. **(3)**

```
int a, *p;

p = ( int * ) malloc ( 50 * sizeof(int) );

for (a = 0; a < 50; a++) *( p + a ) = a + 1;
```

(c) Fill in the blanks to complete the following program that calculates the sum as well as the sum of the squares of an array storing ten floating-point numbers. **(5)**

```
#include <stdio.h>
void fun ( double A[], double *x, double *y )
{
    int i;
    for (i=0; i<10; i++) {

        *x += A[i];

        *y += A[i] * A[i] ;
    }
}
int main()
{
    int i;
    double A[10], sum = 0, sumSqr = 0;
    for (i=0; i<10; i++) {
        printf("Enter element %d:  ", i+1);
        scanf("%lf", &A[i]);
    }

    fun ( A , &sum , &sumSqr );
    printf("The sum of the elements of A is %lf\n", sum);
    printf("The sum of the squares of the elements of A is %lf\n", sumSqr);
    return 0;
}
```

Use the page for rough work