Programming and Data Structures (Autumn 2023–24)

Full marks = 30 **Class Test 2** **Time = 1 hour**

Answer all. Write the answers in the blanks or boxes only.

NAME: _____ ROLL NO.: _____ SEC.: ____

1. Recall that Fibonacci numbers are recursively defined as:

   $$f(0) = 0,\ f(1) = 1,\ f(n) = f(n-1) + f(n-2)\ \forall\ n \geq 2.$$

   Define a new sequence $g(n) = 2f(n) + 3\ \forall\ n \geq 0$. The following code computes $g(n)$ with $n(> 0)$ as input. Fill up the blanks. [6 marks]

```
#include <stdio.h>
int g(int n){

  if _____________________ return _____________; // 1+1 marks

  if _____________________ return _____________; // 1+1 marks

  return ____________________________________; // 2 marks
}

int main(){
  int n;
  printf("Enter n (>0): "); scanf("%d", &n);
  printf("g(n) = %d\n", g(n));
  return 0;
}
```

Answer:

```
if (n == 0) return 3;
if (n == 1) return 5;
return g(n-1) + g(n-2) - 3;
```

2. Fill in the blanks for the following C program that takes a string as input, constructs another string as the uppercase version of the input string concatenated to itself, and prints it. The original input string is left unchanged. For example, if the input string is `iitKgp`, then the new string is `iitKgpIITKGP`. Note that `toupper()` is a function that returns the corresponding uppercase letter if its argument is a lowercase English letter, and returns the same value otherwise. [6×1 marks]

```
#include <stdio.h>
#include <string.h> // for strlen()
#include <stdlib.h> // for malloc() and free()
#include <ctype.h>  // for toupper()

void concatuppercase(char *s1); // Function prototype

int main() {
```

```
  char s1[51];
  printf("Enter a string (at most 50 characters): ");
  scanf("%s", s1);
  concatuppercase(s1);
  return 0;
}

void concatuppercase(char *s1) {
  unsigned s1_len = strlen(s1);
  char *s2; // s2 will contain final string

  s2 = (char *) malloc (_____________________________ + 1);
  strcpy(s2, s1); // copy s1 to s2

  s2 += _________________; /*position s2 at end of string*/
  while (1) {
    *s2 = toupper(________); /*write uppercase letter*/

    if (!*s2) ____________________;
    s1++; s2++;
  }

  ____________ -= 2 * s1_len; /*reposition s2*/
  printf("The new string is= %s\n", s2);

  free(_________);
  return;
}
```

Answer:

```
s2 = (char *) malloc(2 * s1_len + 1);
s2 += s1_len;
*s2 = toupper(*s1);
if (!*s2) break;
s2 -= 2 * s1_len;
free(s2);
```

3. Write the output of the following program. $3+3=6$ marks

```
#include <stdio.h>
void transpose1 (int x[][4], int n){
  int p, q, t;
  for (p=0; p<n; p++)
    for (q=p; q<n; q++){
      t = x[p][q];
      x[p][q] = x[q][p];
      x[q][p] = t;
    }
}

void transpose2 (int x[][4], int n){
```

```
  int p, q, t;
  for (p=0; p<n; p++)
    for (q=0; q<n-p; q++){
      t = x[p][q];
      x[p][q] = x[n - q - 1][n - p -1];
      x[n- q -1][n - p -1] = t;
    }
}

int main(){
  int a[4][4] = {1, 2, 3, 4, 5, 6, 7, 7, 5, 6, 7, 8, 7, 6, 5, 4};
  int p, q;
  transpose2 (a, 4);
  transpose1 (a, 4);

  printf("The Transposed Matrix is:\n");

  for (p=0; p<4; p++){
    for (q=0; q<4; q++)
      printf ("%d  ", a[p][q]);
    printf ("\n");
  }
  transpose2 (a, 4);

  printf("The Super Transposed Matrix is:\n");
  for (p=0; p<4; p++){
    for (q=0; q<4; q++)
      printf ("%d  ", a[p][q]);
    printf ("\n");
  }
  return 0;
}
```

Answer:

```
The Transposed Matrix is:
4 5 6 7
8 7 6 5
7 7 6 5
4 3 2 1
The Super Transposed Matrix is:
1 5 5 7
2 6 6 6
3 7 7 5
4 7 8 4
```

4. Let $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$ be two d-dimensional vectors. Each x_i or each y_i can be a real number or an integer. The *Euclidean distance* between x and y is defined as $\sqrt{(x_1 - y_1)^2 + \cdots + (x_d - y_d)^2}$. Fill up the blanks in the following program to compute the Euclidean distance between two 5-dimensional vectors given in `vec1` and `vec2`. $2+2+1+1 = 6$ marks

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

```
#include <stdlib.h>
#include <math.h>

float EuclidDist(int **ptr){
  float sum=0;
  for(int i=0; i<5; i++){

    sum += _____________________________;
  }
  return sqrt(sum);
}

int main() {
  int **ptr, i;
  int vec1[5] = {1, 2, 3, 4, 5};
  int vec2[5] = {11, 12, 13, 14, 15};

  // allocate memory and assign arrays

  ptr = ____________________________________________________ ;

  ptr[0] = _____________________; ptr[1] = _____________________;

  printf("Euclidean distance: %f", EuclidDist(ptr));
}
```

Answer:

Blank 1: `(ptr[1][i] - ptr[0][i])*(ptr[1][i] - ptr[0][i])`
Blank 2: `(int **)malloc(2*sizeof(int *))`
Blank 3: `ptr[0] = vec1`
Blank 4: `ptr[1] = vec2`

5. The following program first takes in as input the number (≤ 100) of student records, and then the individual fields of each record. It then determines the first occurrence of the top scorer in the list of students and displays the corresponding details. Fill up the blanks. 6×1 marks

```
#include <stdio.h>
int main(){
  typedef struct acad_record{
    char roll[10];
    float score;
  } student;

  _____________________ s[100]; // Declare an array of students

   int i, index, n;
   float max = 0;
   printf("\nEnter the number of student records: ");
   scanf("%d", &n);
```

```
   printf("\nEnter the details of students: ");
   for(i=0; i<n; i++){
     printf("\nEnter roll no. of Student %d: ", i+1);

     scanf(_____________________________________________);

     printf("\nEnter score of Student %d: ", i+1);

     scanf(_____________________________________________);
   }
  for(i=0;i<n;i++){ // Determine the top scorer

    if(_____________________________________________){
      max = s[i].score;
      index = i;
    }
  }
  printf("\nThe first-occurred top scorer:\n");

  printf("Roll: _____________________________________);

  printf("Score: _____________________________________);

  return 0;
}
```

Answer:

```
struct acad_record s[100] (OR) student s[100];
scanf("%s",s[i].roll);
scanf("%f",&s[i].score);
if(s[i].score > max)
printf(Roll : %s", s[index].roll);
printf(Score : %f", s[index].score);
```