CS11001 Programming and Data Structures, Autumn 2014–2015

Class Test 2

Date: 28–October–2014 Time: 7:00–8:00pm Maximum Marks: 20

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

Write your answers in the question paper itself. Be neat and tidy.
Answer all questions. Not all blanks carry equal marks.

1. What are printed by the following programs. **($1\frac{1}{2} \times 6$)**

(a)
```
main ()
{
   int A[5] = {1, 2, 3, 5, 8}, *p;
   p = A;
   printf("%d,%d", p[1]+2, (p+1)[2]);
}
```

4,5

(b)
```
main ()
{
   int i, j, A[10][20], (*p)[20];
   for (i=0; i<10; ++i)
      for (j=0; j<20; ++j)
         A[i][j] = 3*i - 2*j;
   p = A;
   printf("%d,%d,%d",
           (**p)+5, **(p+5), *(*p+5));
}
```

5,15,-10

(c)
```
main ()
{
   int A[3][3]={{1,0,0},{0,1,0},{0,0,1}};
   int i, j;
   for (i=0;i<3;++i) for (j=0;j<3;++j) {
      if (i > 0)
         A[i][j] += A[i-1][j];
      else if (j > 0)
         A[i][j] += A[i][j-1];
   }
   for (i=0;i<3;++i) {
      for (j=0;j<3;++j)
         printf("%d,", A[i][j]);
      printf("\n");
   }
}
```

1,1,1,
1,2,1,
1,2,2,

(d)
```
int f ( int i )
{
   return (i*i + 3) % 10;
}
int main ()
{
   int i, j, A[10];
   A[0] = i = 0;
   while (1) {
      j = f(i);
      A[j] = A[i] + j;
      if (A[j] > 20) break;
      i = j;
   }
   printf("%d,%d", A[i], A[j]);
}
```

14,21

(e)
```
typedef struct {
   char *name;
   int age, ht;
} human;
void hfn ( human A )
{
   A.name[0] += 'A' - 'a';
   A.age += 3; A.ht += 5;
}
int main ()
{
   char A[20] = "computer";
   human H = {A, 20, 180};
   hfn(H);
   printf("%s,%d,%d",H.name,H.age,H.ht);
}
```

Computer,20,180

(f)
```
main ()
{
   char A[10] = "14FB61R23";
   int i, sum = 0;
   for (i=0; A[i]; ++i) {
      if (A[i] < '0') continue;
      if (A[i] > '9') continue;
      sum += A[i] - '0';
   }
   printf("sum = %d", sum);
}
```

sum = 17

2. A digital image is stored as a two-dimensional array of pixel values. In this exercise, we deal with gray-scale images, so each pixel is an integer between 0 and a maximum level (like 255 or 65535), and stands for a shade of gray. Let us use the following structure to store an image.

```
typedef struct {
   int nrows, ncols; /* the vertical and horizontal dimensions */
   int pixel[MAXSIZE][MAXSIZE]; /* two-dimensional array of pixels */
} image;
```

(a) Write a function that, upon the input of an image I and a pair of indices i and j, returns the i, j-th pixel of I. If i, j is not a valid index in the image, the function should return a negative (that is, invalid) value. **(3)**

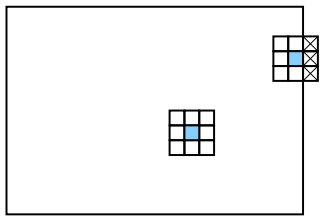
```
int pixval ( image I, int i, int j ) {

   if ((i >= 0) && (i < I.nrows) && (j >= 0) && (j < I.ncols))
      return I.pixel[i][j];
   return -1;

}
```

(b) We are given an image I. We plan to write a function to blur the image as follows.

We use a 3×3 mask around every pixel (i, j) with (i, j) at the center of the mask. The new (i, j)-th pixel value in the blurred image is the average of the nine old pixel values in the 3×3 mask. If the (i, j)-th pixel is at the boundary of the image, then the entire 3×3 mask does not reside inside the image. In that case, only the values of those pixels that reside inside the image are averaged. The adjacent figure shows the mask. The invalid mask pixels for a pixel at the boundary are crossed.



The function below implements this blurring algorithm. Fill out the missing details. For computing the new value for each pixel, `k` stores the number of valid pixels inside the mask. For averaging, divide by `k`. **(7)**

```
image blur ( image I )
{
   image J; /* We work on a local image */
   int i, j, k, t, u, v;
   /* Initialize the dimensions of J (same as those for I) */

   J.nrows = _____I.nrows_____ ; J.ncols = _____I.ncols_____ ;
   for (i = 0; i < J.nrows; ++i) {
      for (j = 0; j < J.ncols; ++j) { /* Calculate the blurred (i,j)-th pixel */

         J.pixel[i][j] = ____0____ ; k = ____0____ ; /* Initialize */
         for (u=-1; u<=1; ++u) for (v=-1; v<=1; ++v) { /* Consider all mask points */

            if ((t = pixval(__I__, __i+u__, __j+v__)) __>= 0__ ) {
               /* Update the pixel value and the count k */

               _____++k_____ ; ____J.pixel[i][j] += t____ ;
            }
         }
         J.pixel[i][j] = __(double)(J.pixel[i][j])/(double)k + 0.5__ ; /* Average */
      }
   }
   return J;
}
```

(c) Suppose that in the `main()` function, an `image` variable `I` stores an image. You want to store the blurred image in `I` itself. How do you call `blur()` for this task? **(1)**

`I = blur(I);`