令和５年度

千葉大学先進科学プログラム入学者選考課題

課題論述

情報

（9:00－12:00）

注意事項

1. この冊子は，監督者から解答を始めるよう合図があるまで開いてはいけません。
2. 問題冊子に印刷または製本の不具合がある場合は，手を上げて申し出てください。
3. 問題すべてに解答してください。
4. 解答用紙は，課題ごとに解答用紙を分けて使用してください。解答用紙は何枚使用してもかまいません。すべての解答用紙に受験番号を必ず記入してください。
5. 検査室に用意してある資料は自由に使用してかまいません。ただし，諸君が持参した教科書，参考書，ノート，パソコンなどの使用は禁止します。
6. 携帯電話やスマートフォン等の電子機器はすべて電源を切り，カバンにしまってください。
7. その他，監督者の指示に従ってください。

問題

1

以下の問いに答えなさい。

問1 次のプログラムを実行したときの標準出力を求めなさい。

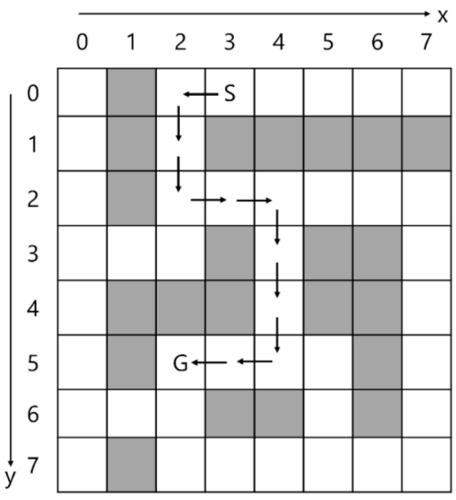
```
#include <stdio.h>
#include <stdlib.h>

#define N 2
#define D 4
int array[N][N];
int dx[D] = {1,0,-1, 0};
int dy[D] = {0,1, 0,-1};
int count = 0;

void init_array(void)
{
   int x, y;

   for(y = 0; y <N; y++)
       for(x = 0; x <N; x++)
          array[y][x] = 0;
}

void print_array(void)
{
   int x, y;

   for(y = 0; y < N; y++){
       for(x = 0; x < N; x++)
         printf("%3d",array[y][x]);
       printf("¥n");
   }
}
/*右上に続く*/
```

```
void try(int x, int y)
{
   int i;

   if (array[y][x]!=0)
       return;
   if (x<0 || x>=N || y<0 || y>=N)
       return;

   count++;
   array[y][x] = count;

   if (count == N*N)
       print_array();
   else
       for(i=0; i<D; i++)
          try(x+dx[i], y+dy[i]);
}

int main(void)
{
   int x=0
   int y=0;

   init_array();
   try(x,y);

   return 0;
}
```

問2　問1のプログラムの try 関数の最後に，array[y][x]=0;　count--;を次に示すように追加する。この追加部分以外の変更はないとする。この新しい try 関数を用いてプログラムを実行したときの標準出力を求めなさい。

```
(01) void try(int x, int y)
(02) {
(03)     int i;
(04)
(05)     if (array[y][x]!= 0)
(06)         return;
(07)     if (x<0 || x>=N || y<0 || y>=N)
(08)         return;
(09)
(10)     count++;
(11)     array[y][x] = count;
(12)
(13)     if (count == N * N)
(14)         print_array();
(15)     else
(16)         for(i=0; i<D; i++)
(17)           try(x+dx[i], y+dy[i]);
(18)     array[y][x] = 0; /* 追加 */
(19)     count--;          /* 追加 */
(20) }
```

問3　右図に示すような N×N(N≧1)の迷路で，スタート(S)から出発して，上下左右のいずれかの方向に1マスずつ移動しながら，ゴール(G)に到達することを考える。灰色のマス目は壁を表し，通り抜けられないとする。また，一つの経路内で，同じ位置を二度通らないとする(例えば，ぐるぐる巡回しない)。



スタート(S)からゴール(G)までの経路をすべて列挙するプログラムを，問2の try 関数を変更して作成しなさい。ただし，上図の迷路だけでなく，すべての N×N の迷路に対して動作するように作成しなさい。解答には，すべてのプログラムを書く必要はなく，変更した行が分かるように記述してあればよい。さらに，なぜそのように変更したのか説明しなさい。作成するプログラムでは，スタート(S)，ゴール(G)，壁(灰色マス目)の位置は，下図のように大域変数(グローバル変数)で与えてあるとする。

```
/*問1のプログラムの 9 行目 int count=0;の下に追加 */
/* S の位置 */
int sx = 3; /* x 座標 in (0,1,…,N-1) */
int sy = 0; /* y 座標 in (0,1,…,N-1) */

/* G の位置 */
int gx = 2; /* x 座標 in (0,1,…,N-1) */
int gy = 5; /* y 座標 in (0,1,…,N-1) */

/* 壁の位置の2次元配列：壁があるとき1，ないとき0*/
int wall[N][N] = { {0,1,0,0,0,0,0,0},
                   {0,1,0,1,1,1,1,1},
                   {0,1,0,0,0,0,0,0},
                   {0,0,0,1,0,1,1,0},
                   {0,1,1,1,0,1,1,0},
                   {0,1,0,0,0,0,1,0},
                   {0,0,0,1,1,0,1,0},
                   {0,1,0,0,0,0,0,0}};
```

問4　問3の `try` 関数をさらに変更し，スタート(S)からゴール(G)までの最短経路を効率的に求めたい。無駄な探索を少しでも減らす工夫を加えたうえで，最短経路を求められるようにプログラムを変更しなさい。必要であれば，大域変数(グローバル変数)を追加してもよい。さらに，なぜそのように変更したのか説明しなさい。

2

問1－1 次に示すプログラムにおいて，(07)行目の count_max の値を増やすほど，プログラムの出力は，ある値に近づくことが期待される。その値は何か答えなさい。なお，rand()は，0から RAND_MAX までの乱数(整数)を返す。 RAND_MAX は処理系により異なるが，一般には，32767以上の非常に大きな整数である。

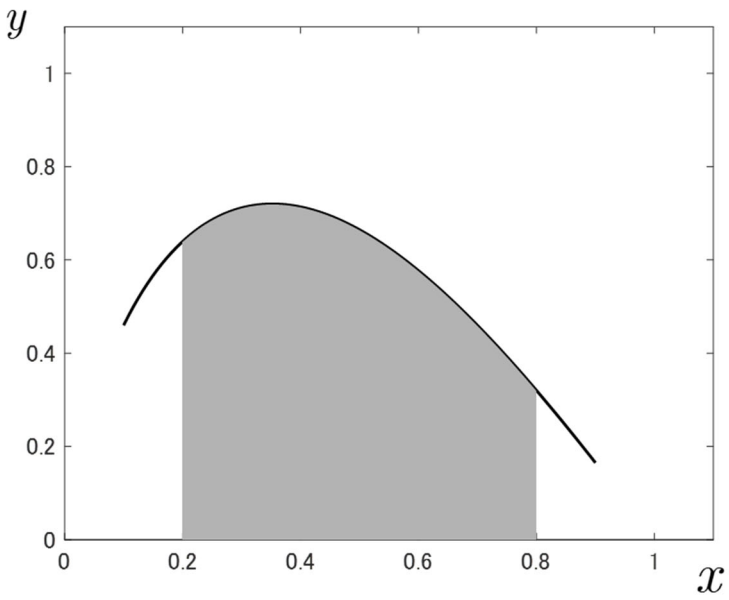
```
(01) #include<stdio.h>
(02) #include<stdlib.h>
(03) #include<time.h>
(04)
(05) int main(void)
(06) {
(07)     int count_max = 1000000;
(08)     int count_in = 0;
(09)     srand((unsigned)time(NULL)); /* 乱数の種を時刻で初期化 */
(10)
(11)     for (int i=1; i<count_max+1; i++){
(12)         double x = rand()/(double)RAND_MAX;
(13)         double y = rand()/(double)RAND_MAX;
(14)         if (x*x+y*y < 1.0) count_in += 1;
(15)     }
(16)
(17)     printf("%lf¥n", 4.0*count_in/count_max);
(18)     return 0;
(19) }
```

問1－2　下図の灰色部分は，曲線 $y = -2\log(x)\sin(x)$ の $0.2 \leqq x \leqq 0.8$ の領域である。この灰色部分の面積の近似値を求めるプログラムを作りたい。問1－1のプログラムにおいて，面積の近似値を出力できるように，(14)行目の条件を改良しなさい。なお，その際，log関数やsin関数などを使うために，問1－1のプログラムに，

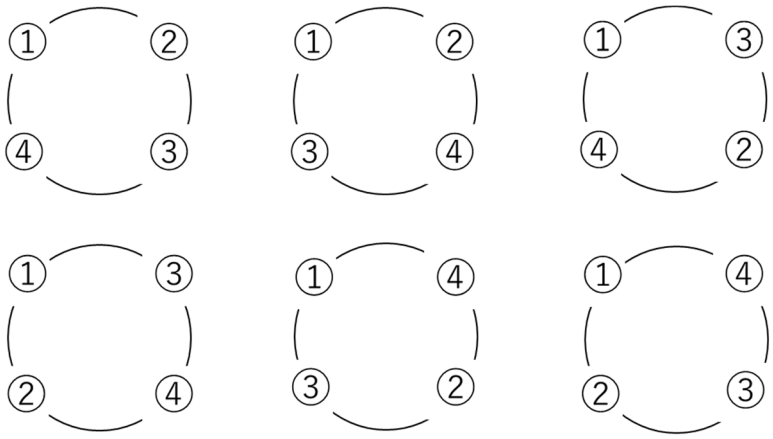
```
#include<math.h>
```

を追加し，問1－1のプログラムの(17)行目を次のように修正するものとする。

```
(17)     printf("%lf¥n", (double) count_in/count_max);
```



問2－1 1からＮまでの自然数を重複がないように円形に並べるとする。例えば，N=4 のとき，1,2,3,4 の並べ方は，次の6通りである（回転して一致する場合は，同一視する）。一般の Ｎ の場合，このような並べ方は，(N-1)!通り存在する。



このような並べ方において，ある2つの数字が隣り合わない条件を満たす並べ方の場合の数を知りたい。例えば，1と3が隣り合わない並べ方は，上の N=4 の場合においては，2通りである。Ｎ が大きいとき，(N-1)!は非常に大きくなり，このような隣り合わない条件が増えると，正確な場合の数を知ることは一般には難しい。しかし，Ｎ が大きいときでも，概ねの場合の数は，プログラムによって，見積もることができる。

まずは，条件に関係なく，ランダムに並べ方を決める。1,2,…,Ｎ の順番に並べた後（配列の初期化），そのなかから，2つをランダムに選んで入れ替えることをＬ回繰り返す。次の空欄を埋めなさい。ただし，次のプログラムでは，1,2,…,Ｎ の代わりに，0,1,…,N-1 の並べ方を考えることにする。必要に応じて，`rand()`を用いなさい。ただし，Ｎ は，`RAND_MAX` を超えないものとする。

プログラム Ａ（並べ方をランダムに決める）：

```
for(j=0; j<N; j++) c[j]=j; /* 配列の初期化 */
for(j=0; j<L; j++){     /* 配列 c の要素の入れ替えを L 回繰り返す。 */
```

空欄

```
}
```

問2－2 問2－1のプログラム A を用いて，次のようなプログラムを作り，count_in/count_max を求めたい。count_max を大きくとるほど，比 count_in/count_max と(N-1)!の積は，一定の値に近づくことが期待される。当然，条件が多すぎると，条件を満たす配列が存在せず，count_in が0になることもある。

```
int count_max = 1000000 /* 条件に関係なくランダムに作成する配列の数 */
int count_in = 0; /* 条件にあう配列の数 count_in の初期化 */
for(i=0; i<count_max; i++){
   count_in += 1;

   プログラム A(並べ方をランダムに決める)

   プログラム B(プログラム A で作成した並べ方が条件に合うかどうか調べる)

}
```

以下，プログラム B を作成する。隣り合わない数字の組合せが重複なく M 個あるとする。

(a[0],b[0]),…,(a[M-1],b[M-1]) (a[j]<b[j]とする)

これらは，配列 a，b として与えられている。問2－1で作成した配列 c が，条件(この場合，隣り合わない数字の組合せ)を満たした並べ方かどうかをプログラム B で調べる。下の空欄を埋めなさい。

プログラム B(プログラム A で作成した並べ方が条件に合うかどうか調べる):

```
int flag=1;  /* 条件に合う配列として flag=1 を立てる。*/
for(j=0; j<M && flag; j++){
   for(k=0; k<N; k++){
      if( [ 空欄 ] ){
         count_in -= 1; /* 条件に合わない配列だったので，count_in を下げる。 */
         flag=0;     /* 条件に合わない配列だったので，flag=0 にする。  */
         break;      /* 条件に合わない配列だったので，for 文を抜ける。  */
      }
   }
}
```

3

各画素が0から255の256階調の明るさで表現されるpgm形式のグレースケール画像(以下，pgm画像)を考える。pgm画像では，ヘッダに続き，各画素の階調値が格納されている。画像の左上を原点(0,0)として，水平方向の右向きをx方向の正の向き，垂直方向の下向きを y 方向の正の向きとする。各画素の値は，原点を起点として，x方向の正の向きに走査され，各行の最右画素に達すると，1 行下の最左画素に移動する。サイズが128×128画素以下のpgm画像に対して，下図のように左右反転を施すことを考える。このプログラムが以下のように与えられるとき，次の問いに答えなさい。以下では，もとの画像の画像番号を 0，左右反転された画像の画像番号を 1 とする。

本プログラムでは，以下で定義される `unsigned char` 型の大域変数(グローバル変数)の3次元配列

```
unsigned char image [2][128][128];
```

を用いて，各画素の階調値を 3 次元配列に格納する。左の要素から順に，画像番号(0から開始される)，x座標，y座標を表す。また，本プログラムでは，`pgmlib.h` というヘッダファイルを用いる。この内部では `stdlib.h` と `string.h` がインクルードされている。 本プログラムに用いられる関数および変数については，`pgmlib.h` において以下のとおり定義されている。

```
void load_image( int n, char name[] );
```

階調画像を入力する関数

n:画像番号，name[]:ファイル名(""のときはキーボード入力)

```
void save_image( int n, char name[] );
```

階調画像を出力する関数

n:画像番号，name[]:ファイル名(""のときはキーボード入力)

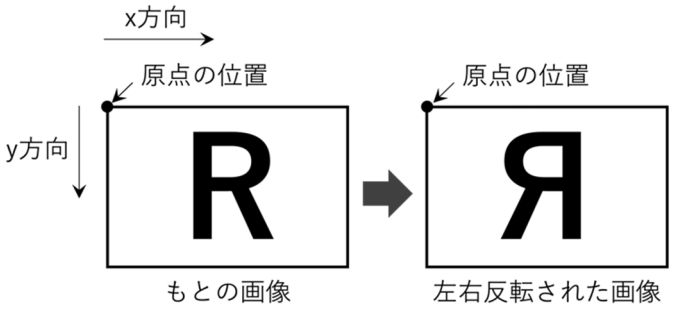
```
int width[2];
```

画像の幅を格納する配列

```
int height[2];
```

画像の高さを格納する配列

なお，上記の 3 次元配列 `image` についても同様に，`pgmlib.h` に定義されていることとする。



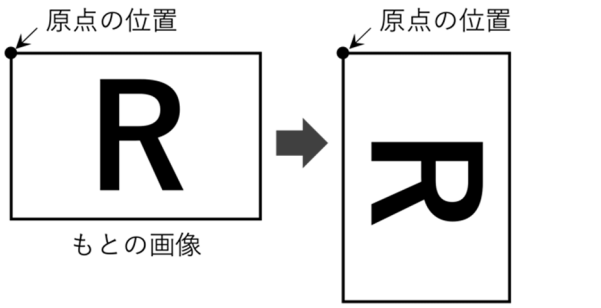
```
(1)  #include<stdio.h>
(2)  #include"pgmlib.h"
(3)
(4)  void img_proc( int  p1, int p2 );
(5)
(6)  int main(void)
(7)  {
(8)      load_image( 0, "" );
(9)      img_proc( 0, 1 );
(10)     save_image( 1, "" );
(11)     return 0;
(12) }
(13)
(14) void img_proc( int p1, int p2)
(15) {
(16)     int x,y;
(17)
(18)     width[p2]=width[p1];
(19)     height[p2]=height[p1];
(20)     for(y=0;y<height[p1];y++){
(21)         for(x=0;x<width[p1];x++)
(22)             image[p2][x][y] = image[p1][width[p1]-1-x][y];
(23)         }
(24) }
```

問1 画像サイズが，64×32画素のとき，この pgm 画像は何バイトであるか答えなさい。ただし，ヘッダの情報は考慮しない。

問2 上記のプログラムを上下反転のプログラムに変更する場合，(22)行目をどのように変更すればよいか答えなさい。ただし，その他の行は変更してはならない。

問3 256階調の画像の場合，例えば，0の階調値を255に，1の階調値を254に変更する処理を階調反転という。上記のプログラムを階調反転のプログラムに変更する場合，(22)行目をどのように変更すればよいか答えなさい。ただし，その他の行は変更してはならない。

問4　下図のように，もとの画像を右回り（時計回り）に90度回転させることを考える。上記のプログラムをこの回転のプログラムに変更する場合，(18)，(19)，(22)行目をどのように変更すればよいか答えなさい。ただし，その他の行は変更してはならない。



右回りに90度回転された画像

問5 上下/左右反転については，その両方向に反転，いずれか一方向のみに反転，あるいは，いずれの方向にも反転しないというパターンが存在する。階調反転については，反転する，あるいは，しないというパターンがある。さらに，回転については，時計回りに，90度，180度，270度，360度（あるいは0度）の回転を施すパターンを想定する。例示している「R」が記された画像について，もとの画像に，上下/左右反転，階調反転，回転の三つの処理を 1 回ずつ実行したとき，処理後の画像のパターンは何種類あるか答えなさい。ただし，各処理において，画像に変化が生じない場合（反転しない，360 度回転するという処理が選択された場合）もその処理は実行されたものとし，処理後の画像のパターンにカウントする。解答用紙には，計算過程も示しなさい。

問6 問1のサイズをもつ画像を16×16画素のブロックに分割し，各ブロックの位置を入れ替える処理を施す。このとき，処理後の画像のパターンは何種類あるか答えなさい。ただし，互いが同一となるブロックは存在しないものとする。解答用紙には，計算過程も示しなさい。

4

Python のプログラムに関する次の記述を読んで設問に答えなさい。

Python で2次元リストを生成する場合，2重にしたリスト表記演算子[]のなかにコンマで区切って要素を並べる方法（コード1）と，リスト内包表記を用いる方法（コード2）がある。コード1とコード2とでは初期化された2次元リスト `data` の要素数は同一であり，また要素の値もいずれも全て0となる。なお，コード2では変数 `width` と `height` の値を変更するだけで要素数を容易に変更できる点が特徴である。

```
data = [
    [0, 0, 0, 0],
    [0, 0, 0, 0],
    [0, 0, 0, 0],
    [0, 0, 0, 0],
]
```

コード 1

```
width = 4
height = 4
data = [[0] * width for i in [0] * height]
```

コード 2

デジタル画像とは，キャンバス上に縦横の格子状に並んだ画素に値を記録し，その値に応じた色に置き換えることで表された画像のことをいう。簡単のために，画素の値が0のときに黒，1のときに白を表し，画素の値としてこれ以外の値は用いないものとする。

問1 まず，図1に示すデジタル画像を考える。このデジタル画像におけるキャンバスの解像度，すなわち幅と高さはそれぞれ4画素であり，黒色と白色の画素がそれぞれ図1のように配置されている。なお，各画素の境界を示す格子模様を付加して表示してある。

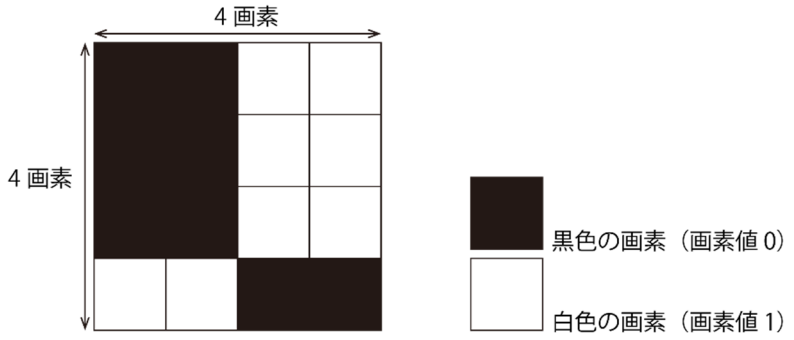


図 1

このデジタル画像のキャンバスは2次元リスト `canvas1` を用いてコード3のように表記できる。

```
canvas1 = [
    [0, 0, 1, 1],
    [0, 0, 1, 1],
    [0, 0, 1, 1],
    [1, 1, 0, 0],
]
```

コード 3

同様にして図2に示すデジタル画像のキャンバスを2次元リスト `canvas2` を用いて表すとき，空欄1を記述しなさい。記述は複数行にわたっても構わない。ただし，モジュールのインポートは行わないこと。

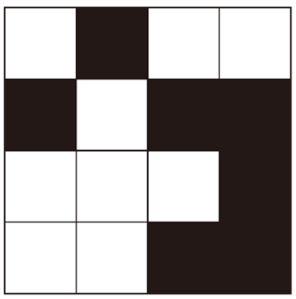
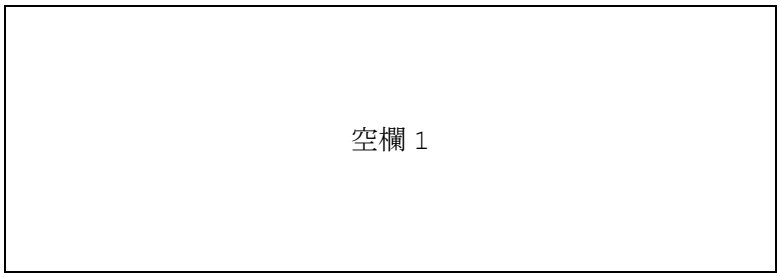


図 2

```
canvas2 = [

]
```



問2　次のコード4に示す関数 all_white は，まず，与えられた引数(width，height)にもとづいて任意の解像度のキャンバス(幅 width 画素，高さ height 画素)を表す 2 次元リストを作成する。このときこの2次元リストの全要素の値は0(つまり全画素が黒)である。次に，キャンバスの全画素値に1を代入して白色を指定し，最後にそのキャンバス canvas を返却する。

```
def all_white(width, height):

  # 幅 width 画素，高さ height 画素でキャンバスを作成
  canvas = [[0] * width for i in [0] * height]

  # 全画素を白色（画素値 1）に設定
  for h in range(height):
    for w in range(width):
      canvas[h][w] = 1

  return canvas
```

コード 4

この関数を実行して得られたキャンバス canvas を画像として表示すると図3のように全ての画素が白色である。なお，関数の引数(width，height)にはそれぞれ8を代入したときの結果を示した。

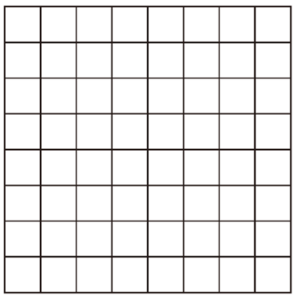


図 3

このことを参考にして，図4のような市松模様のデジタル画像を示すキャンバスを返却する関数 checker を作成する。模様の規則性を利用して，空欄2を記述しなさい。記述は複数行にわたっても構わない。ただし，モジュールのインポートは行わないこと。なお，キャンバスの解像度(幅 width，高さ height)は任意の値が与えられるものとして解答しなさい。

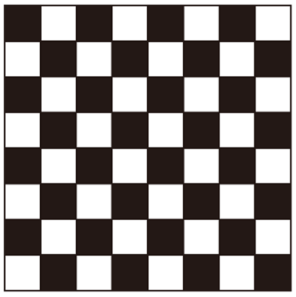


図4

```
def checker(width, height):

  # 幅 width 画素，高さ height 画素でキャンバスを初期化
  canvas = [[0] * width for i in [0] * height]

  for h in range(height):
    for w in range(width):

      空欄 2

  return canvas
```

問3　デジタル画像はキャンバスの縦，横の大きさを増やしていくにつれてより精細に図形を表現することができるようになる。そこで，図5に示すような中心の画素位置が(cx, cy)，半径が r 画素の白色の円を示すキャンバスを返却する関数 circle を作成する。このときの空欄3を記述しなさい。

なお，図形の境界とちょうど重なる位置にある画素の色は白色にすること。記述は複数行にわたっても構わない。ただし，モジュールのインポートは行わないこと。また，キャンバスの解像度（幅 width，高さ height）は任意の値が与えられるものとして解答しなさい。

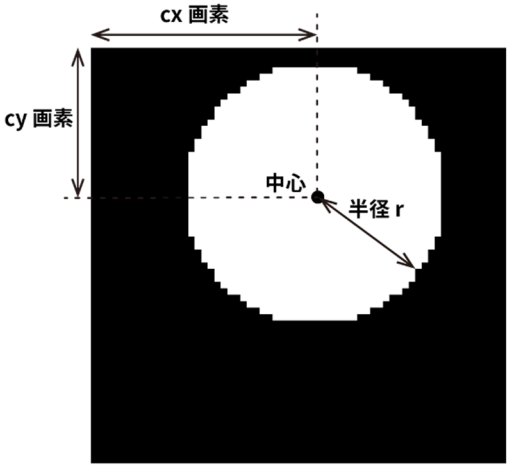


図 5 なお，各画素の境界を示す格子模様は表示を省略した。

```
def circle(width, height, cx, cy, r):

  # 幅 width 画素，高さ height 画素でキャンバスを初期化
  canvas = [[0] * width for i in [0] * height]

  for h in range(height):
    for w in range(width):

      空欄 3

  return canvas
```

問4　図6に示すような白色のリングを示すキャンバスを返却する関数 `ring` を作成する。なお，リングは中心の画素位置が`(cx, cy)`，半径が `r` 画素であり，リングの太さは `t` 画素とする。このときの空欄4を記述しなさい。

なお，図形の境界とちょうど重なる位置にある画素の色は白色にすること。記述は複数行にわたっても構わない。ただし，モジュールのインポートは行わないこと。また，キャンバスの解像度（幅 `width`，高さ `height`）は任意の値が与えられるものとして解答しなさい。

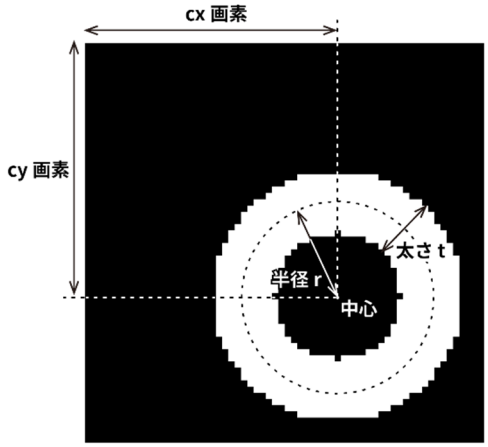


図 6 なお，各画素の境界を示す格子模様は表示を省略した。

```
def ring(width, height, cx, cy, r, t):

  # 幅 width 画素，高さ height 画素でキャンバスを初期化
  canvas = [[0] * width for i in [0] * height]

  for h in range(height):
    for w in range(width):

      空欄 4

  return canvas
```

問5 図7に示すような一定の太さの白色の線分が描かれたキャンバスを返却する line 関数を作成する。なお，線分は 2 点 (xa, ya)，および(xb, yb) を通り，線の太さは t 画素とする。このときの空欄5を記述しなさい。

なお，図形の境界とちょうど重なる位置にある画素の色は白色にすること。記述は複数行にわたっても構わない。ただし，モジュールのインポートは行わないこと。また，キャンバスの解像度(幅 width，高さ height)は任意の値が与えられるものとして解答しなさい。

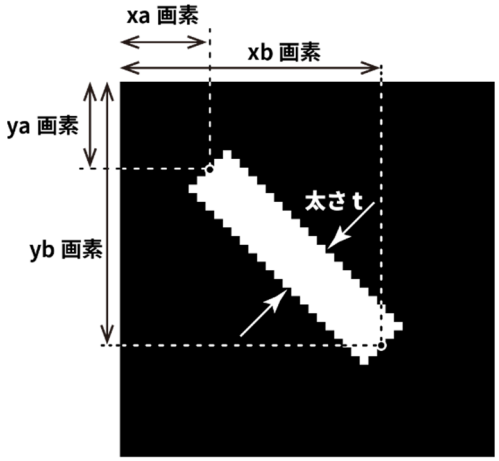


図 7 なお，各画素の境界を示す格子模様は表示を省略した。

```
def line(width, height, xa, ya, xb, yb, t):

  # 幅 width 画素，高さ height 画素でキャンバスを初期化
  canvas = [[0] * width for i in [0] * height]

  for h in range(height):
    for w in range(width):
```

空欄 5