《数据结构》第三次作业

堆栈与递归算法实验

# 一、数制转换：将十进制数转换为十六进制

随机生成1个10000到99999之间的整数，将其转换为十六进制输出，并比较其结果与printf(”%x”, n)是否一致。

要求：

1. 用堆栈实现转换；
2. 随机生成1个10000到99999之间的整数n;
3. 显示要查找的数据n；
4. 显示转换后的十六进制数和printf(”%x”, n)显示结果。

程序代码：

#pragma GCC optimize(1)

#pragma GCC optimize(2)

#pragma GCC optimize(3,"Ofast","inline")

#include<bits/stdc++.h>

#include<regex>

using namespace std;

void conversion(long n, int sys){

if(n>0){

conversion(n/sys,sys);

int r = n%sys;

if(r<=9){

printf("%d",r);

}

else{

printf("%c",r+87);

}

}

}

int main()

{

// 生成longlong随机数不能用rand，方法如下：

mt19937 generator(time(0));

uniform\_int\_distribution<long long> distribution(10000,99999);

long long n = distribution(generator);

cout<<n<<"\n";

int sys = 16;

conversion(n,sys);

printf("\nprintf(\"%%x\",n)显示的十六进制数：%x",n); // 用%%输出%

return 0;

}

运行结果：

第一次：

原始数据：47006

转换后显示数据：b79e

Printf(“%x”)显示数据：b79e

第二次：

原始数据：31025

转换后显示数据：7931

Printf(“%x”)显示数据：7931

第三次：

原始数据：71387

转换后显示数据：116db

Printf(“%x”)显示数据：116db

# 二、迷宫算法(用堆栈或或递归实现)

随机生成42x42的迷宫，要求迷宫中可落脚点和所占比例约为25%。以(1,1)点为起始点，终点为(40,40)，试绘制从起点到终点的路径图。

要求：

1. 显示原始迷宫，以“□”表示可以落脚的点，以“■”表示不可落脚的点；
2. 第0行、第0列、第41行、第41列为不可落脚点；
3. 只显示第1~40行、第1~40列之间的数据；
4. 以“□”表示未尝试的可落脚点，“■”表示不可落脚点，“※”表示路径点，“×”表示尝试过但不能达到目的点的路径图。
5. 按如下方式显示路径序列：（1,1）🡪（1,2）🡪 … 🡪（40,40）；
6. 可以采用堆栈方式或者递归方式实现。

说明：

若目标点不可达，则重新生成随机数，再试一次。

程序代码：

#pragma GCC optimize(1)

#pragma GCC optimize(2)

#pragma GCC optimize(3,"Ofast","inline")

#include<bits/stdc++.h>

#include<regex>

using namespace std;

#define N 40

int maze[N+2][N+2]; // 0不可落脚 1未走到 2已走过 3走过但不能到达目的点的路径

typedef struct StackNode{

int x;

int y;

struct StackNode \*below;

}\*Stack;

void push(Stack \*ps,int x,int y){

Stack top = (Stack)malloc(sizeof(StackNode));

top->x = x;

top->y = y;

top->below = \*ps;

\*ps = top;

}

Stack path = (Stack)malloc(sizeof(StackNode));

bool find\_path(int x, int y, int former\_di){ // former\_di 0上 1右 2下 3左

if(x==N&&y==N){

maze[x][y] = 2;

path->x = x;

path->y = y;

return 1;

}

if(former\_di!=1&&maze[x][y+1]==1){

maze[x][y] = 2;

if(find\_path(x,y+1,3)==1){

push(&path,x,y);

return 1;

}

}

if(former\_di!=2&&maze[x+1][y]==1){

maze[x][y] = 2;

if(find\_path(x+1,y,0)==1){

push(&path,x,y);

return 1;

}

}

if(former\_di!=3&&maze[x][y-1]==1){

maze[x][y] = 2;

if(find\_path(x,y-1,1)==1){

push(&path,x,y);

return 1;

}

}

if(former\_di!=0&&maze[x-1][y]==1){

maze[x][y] = 2;

if(find\_path(x-1,y,2)==1){

push(&path,x,y);

return 1;

}

}

maze[x][y] = 3;

return 0;

}

int main()

{

ios::sync\_with\_stdio(false);

// 填充边界

LABLE1:

for(int i=0;i<N+2;i++){

maze[0][i] = 0;

}

for(int i=1;i<N+2;i++){

maze[i][0] = 0;

maze[i][N+1] = 0;

}

for(int i=1;i<N+1;i++){

maze[N+1][i] = 0;

}

int a = (N+2)\*(N+2);

int b = (N+2)\*(N+2)\*1/4-(2\*N+2)\*2;

srand(time(0));

for(int i=1;i<N+1;i++){

for(int j=1;j<N+1;j++){

maze[i][j] = (rand()%a>=b);

}

}

for(int i=1;i<N+1;i++){

for(int j=1;j<N+1;j++){

switch(maze[i][j]){

case 0: printf("■");break;

case 1: printf("□");break;

}

}

printf("\n");

}

printf("\n");

maze[1][1] = 1;

maze[N][N] = 1;

path->below = NULL;

if(find\_path(1,1,0)==0){

system("cls");

goto LABLE1;

}

for(int i=1;i<N+1;i++){

for(int j=1;j<N+1;j++){

switch(maze[i][j]){

case 0: printf("■");break;

case 1: printf("□");break;

case 2: printf("※");break;

case 3: printf("×");break;

}

}

printf("\n");

}

printf("(%d,%d)",path->x,path->y);

while(path->below!=NULL){

path = path->below;

printf("->(%d,%d)",path->x,path->y);

}

return 0;

}

运行结果：

原始图示：

□□■■■□□■□■■□□□■□■■■■■□■□□□■□□□□■□□■■■□■□

□■□□□□□□□■■□□□■■■■□□□□□□■□□□□□□□■□□□□■□□

□□□□■■□□■□□□□□■□□□■□■□■□■□□□■■□□□□■□□■□□

□□□□■□■□□□■□■■■■□□■□■■■□■■□■■□■□■□□■■■□■

□□■□■□□□■□■□□□□■□□■■□■□□□■□□■■■□■□□□□■□■

□□□□□■■□□□□■□■■■■■■■□□□□□□□■□■■□■□■■□□■■

□■□■■□□■□□□□□□□■□□■■□■□■■□□■■□□□□■■□□■■□

■■□□□■□■□□□□■□□■■□■□□□□■□□□□□■■□□■□□□□■□

□■□□□□■■□□□□■□□□□□□□□■□□□■■□■■■□■■□□■□■□

□□□■□■□■■■□■□□□□□□■■□□□□□□□□□■□■□□□■□■□□

□□□□■□□□■□□■□■□■□■□■■■□■□□□□□□□□□□□□■□□■

□□■□■□□□□■■■■□□□■■□□□■■■■□□□■□□■■■□□■□□□

■□□□□□□□□□□□□□□■■■□□□□□■■■■■□■□□□■■■□□■□

□□□■■□□■■□■■□□■□■□■■□■□□□□■□□■□■□□□□□■□■

□■■□□■□□□■□■□□■■□■■■■■□■□□□□■□□□□■■□□■■□

□□■■■■■□□□□■■□□■■■□□□□■■■□■□□■□■□□□□□□□□

□■■□□□□□□■□□□□□□□■□■□□■■□□■□□□□■□■□□□■■■

■■■■□□□□□□■□□□■□□■■□■□□□□□■□□■□□□■□■□□□■

□□□□■□□■■□□■■■■□□□■□□□□□■■■□□■□■■□□□□□□■

■■□■□□□■■■□■□□■□□□□□□■■□■■■■□□□□□■■□□□□■

■■□□□■□■□□■■■■□□□□■■□□□■□■□□■■□□■■□■■□□□

□■□■□■□■■□■□■■□□□□□□□□■□■□■■□■■□■■□■■■■□

■□□□■□□□□□□□□■■□□■□□□□□■□□■■■□□□■□□□■■■■

□□□■□□□□□□■□■■□□□□■■□□■□■■□■□■□□□■■□□■■□

■□□□□□□■□■□□■■■□□■□□□□□□■■□□□□■□□□□□■□□□

■□□□□■■□□□□■□□■■□□□■■□■□□■□■□■□■■□□■□□□■

□□□□□■□□□□□□□□□■□■■□□□■□□□■■□□■■□■□□□■□■

■□□□■□■□□□□□□■□□□■□□□■■■□□■□□■□■■■□□□□■□

□□■■□■□■■□□■□□□■□■□□□□■□□□■□■□■■□□□■□□□□

□■□■■■■□□□□□□□□□■■■■■■■■□■□□□□□□□■□■■□□■

□□□■■□□□■□■□□■■□□■□□■■□■□□□□□□□■□□□□□■■■

■□■□□□□■■■□□□□□■□■□□□□■□■■□■□□□□□■■□■■■■

□□■□□□□□□□□□■□□■■■□□■■□■■■□■□□□□■□■□□□■■

□□■■□□■■□□□□□■□□□□■□□■□■□□■■□■□■■□■□□■□□

□■□□■□■■■□■□■□■■□□□■□■■■□□■■□□□□□■□□■■■□

□□□□□□■□□□□□■□□□■□■□■■□□■□□□■■■□■□■■■□■□

□□□□□■□■■□□□■■□□■■□□■□■□□□□□□□■□■□□■□■□□

□□□□■□■□□□□□□■□■□□■□□■■■□□□■□□□□□■□□■■□■

■□□□■□■■■■■□□□□□□■□■■□□□■■■■■■■■□■■□□■□■

□□□□■■■■□□■■■□□□□□■■□□■□■■□□□□□□□□□□□□□□

结果图示：

(1,1)->(2,1)->(3,1)->(3,2)->(3,3)->(3,4)->(4,4)->(5,4)->(6,4)->(6,3)->(7,3)->(8,3)->(8,4)->(8,5)->(9,5)->(9,4)->(9,3)->(10,3)->(11,3)->(11,4)->(12,4)->(13,4)->(13,5)->(13,6)->(13,7)->(13,8)->(13,9)->(13,10)->(13,11)->(13,12)->(13,13)->(13,14)->(13,15)->(12,15)->(11,15)->(10,15)->(10,16)->(10,17)->(10,18)->(9,18)->(9,19)->(9,20)->(9,21)->(10,21)->(10,22)->(10,23)->(10,24)->(10,25)->(10,26)->(10,27)->(10,28)->(10,29)->(11,29)->(11,30)->(11,31)->(12,31)->(13,31)->(13,32)->(13,33)->(14,33)->(14,34)->(14,35)->(14,36)->(14,37)->(15,37)->(16,37)->(17,37)->(18,37)->(18,38)->(18,39)->(19,39)->(20,39)->(21,39)->(21,38)->(20,38)->(20,37)->(20,36)->(19,36)->(19,35)->(18,35)->(17,35)->(17,36)->(16,36)->(16,35)->(16,34)->(16,33)->(17,33)->(18,33)->(18,32)->(18,31)->(19,31)->(20,31)->(20,32)->(21,32)->(22,32)->(23,32)->(24,32)->(24,33)->(25,33)->(25,34)->(25,35)->(26,35)->(27,35)->(27,36)->(27,37)->(28,37)->(28,36)->(28,35)->(29,35)->(30,35)->(31,35)->(31,34)->(31,33)->(32,33)->(32,32)->(33,32)->(33,31)->(34,31)->(35,31)->(35,32)->(36,32)->(37,32)->(38,32)->(38,33)->(39,33)->(40,33)->(40,34)->(40,35)->(40,36)->(40,37)->(40,38)->(40,39)->(40,40)

路径图：

※×■■■□□■□■■□□□■□■■■■■×■×××■××××■××■■■□■□

※■□□□□□□□■■□□□■■■■××××××■×××××××■××××■□□

※※※※■■□□■□□□□□■□□□■×■×■×■×××■■××××■××■□□

□□□※■□■□□□■□■■■■□□■×■■■×■■×■■□■×■××■■■□■

□□■※■□□□■□■□□□□■□□■■×■×××■××■■■×■××××■□■

□□※※×■■□□□□■□■■■■■■■×××××××■□■■×■×■■××■■

□■※■■□□■□□□□□□□■□□■■×■×■■××■■××××■■××■■□

■■※※※■□■□□□□■□□■■□■××××■×××××■■××■××××■□

□■※※※×■■□□□□■□□□□※※※※■×××■■×■■■×■■××■×■□

□□※■×■□■■■□■□□※※※※■■※※※※※※※※※■□■×××■□■□□

□□※※■□□□■□□■□■※■□■□■■■□■□□□□※※※×××××■□□■

□□■※■□□□□■■■■×※×■■□□□■■■■□□□■□※■■■××■□□□

■□□※※※※※※※※※※※※■■■□□□□□■■■■■□■※※※■■■□□■□

□□□■■□□■■□■■□□■□■□■■□■□□□□■□□■□■※※※※※■□■

□■■□□■□□□■□■□□■■□■■■■■□■□□□□■□□□□■■□※■■×

□□■■■■■□□□□■■□□■■■□□□□■■■□■□□■□■※※※※※×××

□■■□□□□□□■□□□□□□□■□■□□■■□□■□□□□■※■※※※■■■

■■■■□□□□□□■□□□■□□■■□■□□□□□■□□■※※※■※■※※※■

□□□□■□□■■□□■■■■□□□■□□□□□■■■□□■※■■×※※××※■

■■□■□□□■■■□■□□■□□□□□□■■□■■■■□□※※×■■※※※※■

■■□□□■□■□□■■■■□□□□■■□□□■□■□□■■□※■■×■■※※×

□■□■□■□■■□■□■■□□□□□□□□■□■□■■□■■※■■×■■■■×

■□□□■□□□□□□□□■■□□■□□□□□■□□■■■□□※■×××■■■■

□□□■□□□□□□■□■■□□□□■■□□■□■■□■□■□※※■■××■■□

■□□□□□□■□■□□■■■□□■□□□□□□■■□□□□■□※※※×■□□□

■□□□□■■□□□□■□□■■□□□■■□■□□■□■□■□■■□※■□□□■

□□□□□■□□□□□□□□□■□■■□□□■□□□■■□□■■□■※※※■□■

■□□□■□■□□□□□□■□□□■□□□■■■□□■□□■□■■■※※※×■×

□□■■□■□■■□□■□□□■□■□□□□■□□□■□■□■■□□※■××××

□■□■■■■□□□□□□□□□■■■■■■■■□■□□□□□□□■※■■××■

□□□■■□□□■□■□□■■□□■□□■■□■□□□□□□□■※※※××■■■

■□■□□□□■■■□□□□□■□■□□□□■□■■□■□□□※※■■×■■■■

□□■□□□□□□□□□■□□■■■□□■■□■■■□■□□※※■□■×××■■

□□■■□□■■□□□□□■□□□□■□□■□■□□■■□■※■■□■××■□□

□■□□■□■■■□■□■□■■□□□■□■■■□□■■□□※※×■××■■■□

□□□□□□■□□□□□■□□□■□■□■■□□■□□□■■■※■□■■■□■□

□□□□□■□■■□□□■■□□■■□□■□■□□□□□□□■※■□□■□■□□

□□□□■□■□□□□□□■□■□□■□□■■■□□□■□□□※※■□□■■□■

■□□□■□■■■■■□□□□□□■□■■□□□■■■■■■■■※■■□□■□■

□□□□■■■■□□■■■□□□□□■■□□■□■■□□□□□□※※※※※※※※

# 三、用队列实现求迷宫问题的最短路

用队列算法实现第二题的最短路（要求所用数据与第二题一致）。

程序代码：

#pragma GCC optimize(1)

#pragma GCC optimize(2)

#pragma GCC optimize(3,"Ofast","inline")

#include<bits/stdc++.h>

#include<regex>

using namespace std;

#define N 40

int maze[N+2][N+2]; // Inf不可落脚 -1未走到 其他步数

struct data{

int x;

int y;

struct QueueNode \*prestep; // 路径中的前一点

};

typedef struct QueueNode{

struct data info;

struct QueueNode \*before; // 队列中的上一项

};

typedef struct Queue{

QueueNode \*front;

QueueNode \*rear;

int size;

};

void initQueue(Queue \*pq){

pq->front = (QueueNode\*)malloc(sizeof(QueueNode));

pq->rear=NULL;

pq->size = 0;

}

void push(Queue \*pq,data \*pd){

QueueNode \*front = (QueueNode\*)malloc(sizeof(QueueNode));

front->info = \*pd;

front->before = NULL;

pq->front->before = front;

pq->front = front;

if(pq->rear==NULL) pq->rear = pq->front;

pq->size++;

}

void pop(Queue \*pq){

QueueNode \*temp = pq->rear;

temp = temp->before;

free(temp);

pq->size--;

}

Queue path;

bool all\_next\_step(){

QueueNode \*pq = path.rear;

int x = pq->info.x;

int y = pq->info.y;

if(maze[x][y+1]==-1){

data info;

info.x = x;

info.y = y+1;

info.prestep = pq;

push(&path,&info);

maze[x][y+1] = maze[x][y]+1;

if(x==N&&y+1==N) return 1;

}

if(maze[x+1][y]==-1){

data info;

info.x = x+1;

info.y = y;

info.prestep = pq;

push(&path,&info);

maze[x+1][y] = maze[x][y]+1;

if(x+1==N&&y==N) return 1;

}

if(maze[x][y-1]==-1){

data info;

info.x = x;

info.y = y-1;

info.prestep = pq;

push(&path,&info);

maze[x][y-1] = maze[x][y]+1;

}

if(maze[x-1][y]==-1){

data info;

info.x = x-1;

info.y = y;

info.prestep = pq;

push(&path,&info);

maze[x-1][y] = maze[x][y]+1;

}

path.rear = path.rear->before;

path.size--;

// 本题出队的节点还要回溯用，因此不要用pop释放掉了

return 0;

}

void print\_path(QueueNode \*p){

if(p->info.prestep==NULL) printf("(%d,%d)",p->info.x,p->info.y);

else{

print\_path(p->info.prestep);

printf("->(%d,%d)",p->info.x,p->info.y);

}

}

int main()

{

ios::sync\_with\_stdio(false);

// 填充边界

LABLE1:

for(int i=0;i<N+2;i++){

maze[0][i] = INT\_MAX;

}

for(int i=1;i<N+2;i++){

maze[i][0] = INT\_MAX;

maze[i][N+1] = INT\_MAX;

}

for(int i=1;i<N+1;i++){

maze[N+1][i] = INT\_MAX;

}

int a = (N+2)\*(N+2);

int b = (N+2)\*(N+2)\*1/4-(2\*N+2)\*2;

srand(time(0));

for(int i=1;i<N+1;i++){

for(int j=1;j<N+1;j++){

if(rand()%10>=4) maze[i][j] = -1;

else maze[i][j] = INT\_MAX;

}

}

maze[1][1] = -1;

maze[N][N] = -1;

for(int i=1;i<N+1;i++){

for(int j=1;j<N+1;j++){

if(maze[i][j]==-1){

printf("□");

}

else printf("■");

}

printf("\n");

}

printf("\n");

initQueue(&path);

data info;

info.x = 1;

info.y = 1;

info.prestep = NULL;

push(&path,&info);

maze[1][1] = 0;

while(all\_next\_step()==0){

if(path.size==0){

system("cls");

goto LABLE1;

}

// 队列为空

}

for(int i=1;i<N+1;i++){

for(int j=1;j<N+1;j++){

if(maze[i][j]==-1) printf("□ ");

else if (maze[i][j]==INT\_MAX) printf("■ ");

else printf("%02d",maze[i][j]);

}

printf("\n");

}

print\_path(path.front);

return 0;

}

运行结果：

原始图示：

□□□■□□■□□■■□■□□□□□■□■■□■□■□□□□■□■■□□□□□■

■■□■■□□□■□□□□□■□□□■□■■■■■□□■□□□■■□■□■■□□

□□□□□□□□□■□■□□■■□■□□■■□□□■□■□□□■□■□□■■□□

□■□□□■■■■■■□■□■□□□□□■■■□■■□■■□■■□■□□■□■■

□□■□■□■□■■□■□□■□□■□■□□□□■□□□□■□□□■■□■□■□

□■□□□□■■□□■□□■■□■□□■■■□□□■■■■□□■■■■■□■□□

■■■□□■■■■□□□□■□□□□□□■□□□□■□□■□■□□□□□■□□■

■□■□□■□■□□□■■□□□□□□□□□■□■□■■□□■□□■□□□□■□

□■□□□□■■□□□□□□□■■□□□□□■■■□□■■□□□□□□■□■□□

□■■■□□□■■□□■■■□■□□□■□□□■□■■□□■□□□■□□□□□□

□■■□□□□□□■■□□□■□■□□□□■■□□□□□□□■■■■□□□■■■

■□□■■□□□□□□□□□■■□□□□■□□■□■□■■■□■□■□□■■■■

■□■□□□□□□□■■■□□■■■□■□□■□□□□■□■□□□□■□■□■■

□□□□■□■□□□□■□□□■■■□□□■□■□□■□□□□□□■■■□□□■

□■□■□■■□□■□□□■□□□□□□□□■■■■□□□□■■□■□■□□□□

□□□□□□□■□■■□□■■■□□□■■□■□■□□□□□■■■□□□□■■■

■■□■■■■■■□□■■□□□□■■□■□□□■■□■□■■■□■■□□■□■

■□□□■■□□□■■□□■□□□■□■□□□□□□□□□□□□□□□□■■□■

■■□□□□□■□□■□□□□□□■■□□■■■■□■□■□□□□■□□□□□□

□■■□■■■□■■□■□■■□■□■□■□□□■□■□□□□□□■■□□■□■

□□□□□□□□■■□■□■□□■■■□■■□□□□□■■□□□■■□■□□■□

□■□□□□■■■□□■□□□□□□■■□■□■■□■■□□■□■■□□□□□■

□■■■□□□■■□□■□□■■□□□□□□□■■■□□■■□□■□□■□■□□

■□□■■■□■□□□□□□■■■□□□■□□■□□■■□□■■■□□□□□□□

■■□□□■■■□□■■□□□■□■■□■■□□□■□■■□■■■□□□■□■■

□■■□□□□□■■□■□□■□□□□□□□□□■■□□□□■■□□□□■■■□

□□□■■□□■■□■■□□□□□■□□■□□■■□■□□■■□□□□□□□■■

□□■■□□□■□■■■□□■□□■■□□■□□□□■□□■■□□□■■□□□■

■□□■□■■□■■□■□■■□■□□□□■■■□■■■□□□□□□□□□■□■

■□■□■□□□■□□□■□■□■■□□□□■□□■□■■□■□□□■■□□■□

■□□□■■□■□■■□□□■□□■□□■□■■■□□□□□■■■□□□□□□■

■□□□■□□■□■■□□□□■■■■□□■□■□□□□■■□□■□□■■□□□

■□□□□□■■■□■■□□□□□■□■■■■■□■■□■■■□□□■□□■□■

■■□□■■■□■■■□□■□■□□□■□□■□■■□■■□■■■■■□□□□□

■□■□□□□■□□□■□□□■□■□■□□□■■□■□■□■■□□□□■□■■

□□□□□□□□□□■□□□□□□■□■□□■■□□□■□□□■■■□□□■■■

■■□□□□■□□□□□□■■□□■■■■□■■□■□□■■■□□□■■□□□■

■□□■■□□□□□□□□□□□■□■□□□■■□■□□□□□■□■□□□□■□

■■■□□□□■□■■□□■■■□□□■□□■□□□□□□□□□□□□□□□□□

■□■□□■■■□□■■□■■□□□□■■□□□□□□■■□■□□□□□□■■□

结果图示：

(1,1)->(1,2)->(1,3)->(2,3)->(3,3)->(3,4)->(4,4)->(5,4)->(6,4)->(6,5)->(7,5)->(8,5)->(9,5)->(9,6)->(10,6)->(10,7)->(11,7)->(11,8)->(11,9)->(12,9)->(12,10)->(12,11)->(12,12)->(12,13)->(12,14)->(13,14)->(13,15)->(14,15)->(15,15)->(15,16)->(15,17)->(15,18)->(15,19)->(15,20)->(15,21)->(15,22)->(16,22)->(17,22)->(17,23)->(17,24)->(18,24)->(18,25)->(18,26)->(18,27)->(18,28)->(18,29)->(18,30)->(18,31)->(18,32)->(18,33)->(18,34)->(18,35)->(18,36)->(19,36)->(19,37)->(20,37)->(21,37)->(22,37)->(23,37)->(24,37)->(24,36)->(25,36)->(26,36)->(27,36)->(27,37)->(28,37)->(29,37)->(30,37)->(30,38)->(31,38)->(31,39)->(32,39)->(33,39)->(34,39)->(34,38)->(34,37)->(34,36)->(35,36)->(36,36)->(36,37)->(37,37)->(37,38)->(38,38)->(39,38)->(39,39)->(39,40)->(40,40)

路径图：

000102■1009■1112■■55■5150494849■47■■□■□■□□□□■□■■□□□□□■

■■03■■080910■5655545352■484748■46■■■■■□□■□□□■■□■□■■□□

060504050607080910■56■5453■■46■4445■■504950■□■□□□■□■□□■■□□

07■050607■■■■■■□■54■4645444344■■■48■■□■■□■■□■□□■□■■

0809■07■11■□■■□■5455■4546■42■48474647■□□□□■□□□■■□■□■□

09■09080910■■5352■5253■■44■4241■■■454647■■■■□□■■■■■□■□□

■■■0910■■■■51505152■444342414041■43444546■□□■□■□□□□□■□□■

■□■1011■□■515049■■444342414039404142■46■□■■□□■□□■□□□□■□

□■12111213■■50494847464544■■3938394041■■■□□■■□□□□□□■□■□□

□■■■131415■■5049■■■45■393837■394041■□■■□□■□□□■□□□□□□

□■■151415161718■■232425■□■37363738■■□□□□□□□■■■■□□□■■■

■2425■■161718192021222324■■37363536■3839■□■□■■■53■55■□□■■■■

■23■19181718192021■■■2526■■■34■3637■□□□□■50■52535455■□■59■■

23222120■18■20212223■272627■■■333435■□■□□■484950515253■■■575859■

24■22■26■■2122■242526■2829303132333435■■■■46474849■■54■56■56575859

25242324252627■23■■2627■■■313233■■36■40■4645464748■■■56555455■■■

■■24■■■■■■□□■■35343332■■□■373839■■44■46■■■50■■5354■58■

■262526■■313233■■4039■353433■□■39383940414243444546474849505152■■57■

■■2627282930■3435■393837363534■■4140■■■■43■45■47484950■525354555657

33■■28■■■34■■49■39■■36■□■42■504948■44■464748495051■■5455■57■

3231302930313233■■48■40■3837■■■43■■4847464546■■495051■■60■5657■□

33■31303132■■■4847■414039383940■■45■47■■46■■5150■52■■5958575859■

34■■■323334■■4746■4241■■40414243444546■■■□□■■5453■6160■58■6061

■□□■■■35■474645444342■■■424344■4647■5152■■7978■■■62616059606162

■■□□□■■■4847■■444344■50■■45■■484950■78■■77■■■636261■61■■

□■■□□□□□■■□■4544■484948474647484950■■77767576■■65646362■■■□

□□□■■□□■■□■■4645464748■4847■4950■■55■7574■■67666564636465■■

□□■■□□□■□■■■4746■4849■■4849■51525354■7473■■686766■■656667■

■□□■□■■□■■□■48■■49■51504950■■■54■■■727170696867686766■68■

■□■□■□□□■□□□■□■50■■51505152■5655■77■■72■706968■■6768■□

■□□□■■□■□■■□□□■5152■5251■53■■■7776757473■■■697069686970■

■□□□■□□■□■■□□□□■■■■5253■□■79787776■■7574■7071■■707172

■□□□□□■■■□■■□□□□□■□■■■■■80■■77■■■737271■7776■72■

■■□□■■■□■■■□□■□■□□□■□□■□■■□■■□■■■■■7675747374

■□■□□□□■□□□■□□□■□■□■□□□■■□■□■□■■80797877■75■■

□□□□□□□□□□■□□□□□□■□■□□■■□□□■□□□■■■797879■■■

■■□□□□■□□□□□□■■□□■■■■□■■□■□□■■■□□□■■808182■

■□□■■□□□□□□□□□□□■□■□□□■■□■□□□□□■□■83828182■□

■■■□□□□■□■■□□■■■□□□■□□■□□□□□□□□□□85848382838485

■□■□□■■■□□■■□■■□□□□■■□□□□□□■■□■□□□858483■■86