Exercise 3. Answer Sheet

Student's Name: Tsuyoshi Kumamoto Student's ID: s1250050

Problem 1. *(25 points)* Consider the following adjacency matrix:

0 1 1 1 0 0

1 0 1 0 0 0

0 0 0 0 1 0

0 1 1 0 1 0

0 1 0 0 0 0

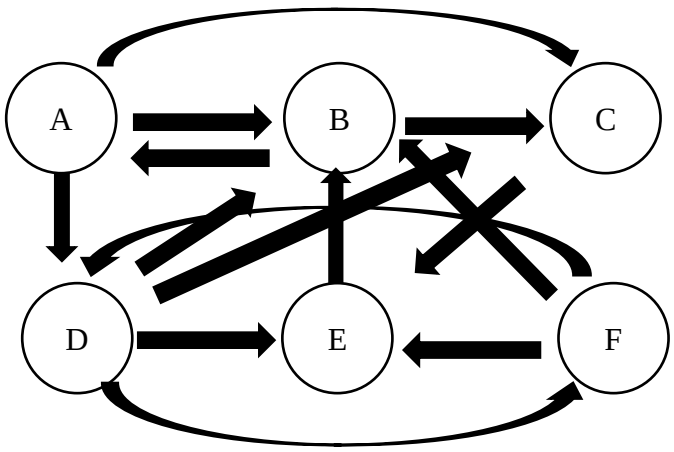
0 0 0 1 1 0

a) Draw a directed graph which corresponds to this adjacency matrix.

Put your answer here.

	a	b	c	d	e	F
A	0	1	1	1	0	0
B	1	0	1	0	0	0
C	0	0	0	0	1	0
D	0	1	1	0	1	0
E	0	1	0	0	0	0
F	0	0	0	1	1	0

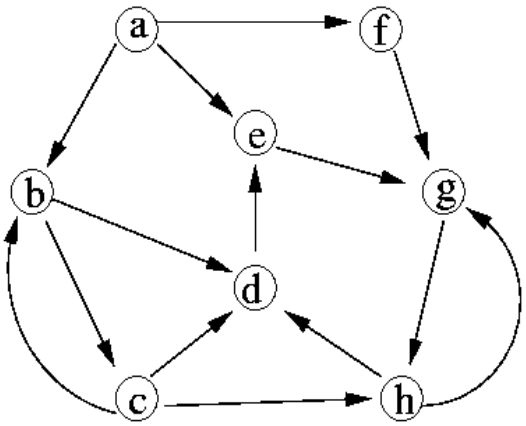
A: b, c, d
B: a, c
C: e
D: b, c, e
E: b
F: d, e



b) Write the adjacency list of the graph from a).

Put your answer here.
A: b, c, d
B: a, c
C: e
D: b, c, e
E: b
F: d, e

Problem 2. *(25 points)* Consider the following graph:

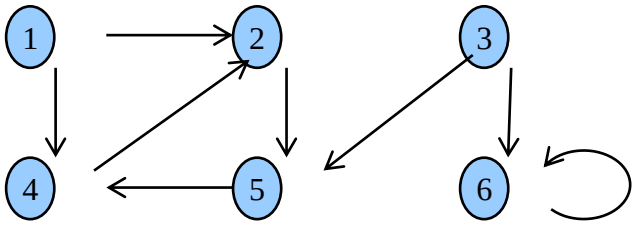


a) Starting from vertex a, in what order the Breath First Search algorithm will traverse the vertices of this graph?
Put your answer here.
a,b,e,f,c,d,g,h

b) Starting from vertex a, in what order the Depth First Search algorithm will traverse the vertices of this graph?

Put your answer here.
a, b, c, h, d,e , g, f

Problem 3. *(50 points)* Based on the pseudo-code for Depth First Search algorithm given at the lecture, write a program implementing it. Given the following graph, starting from node 1, calculate the discovery and finishing time of each node and fill the table starting from node 1. (Don't forget to upload your program!)



Node	1	2	3	4	5	6
Discovery time	1	2	9	4	3	10
Finishing time	8	7	12	5	6	11

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

#define WHITE 0
#define GRAY 1
#define BLACK 2
/*
タイムスタンプ d[v]: v を最初に発見した発見時刻を記録します。
タイムスタンプ f[v]: v の隣接リストを調べ終えた完了時刻を記録します。
*/
typedef struct{
  int id;//要素の中身
  int k;//子供の数
  int *v;//k で動的に確保した子供の ID
  int color;//巡回判定の色
  int dscv;//最初に発見したステップ数が入る
  int fin;//巡回終了したステップ数が入る
}Graph;

Graph *G;
int n;
int **Adj;
int t;


void getDef(){
  int i,j;

  Adj = malloc(sizeof(int*) * (n+1));
  for(i=1;i<=n;i++){
    Adj[i] = malloc(sizeof(int) * (n+1));
    for(j=1;j<=n;j++){
```

```
      Adj[i][j] = 0;
    }
  }
}

void printer(){
  int i, j;

  for(i=1; i<=n; i++){
    printf("G[ID]: %d Discover Time: %d Finished Time: %d\n",G[i].id, G[i].dscv,
G[i].fin);
  }
}

void visit(int id){
  int i;

  //バーテックスの探索を判定する
  G[id].color = GRAY;//探索準備
  G[id].dscv = ++t;
  for(i=1; i<=G[id].k; i++){
    if(G[G[id].v[i]].color == WHITE){//未探索判定
      visit(G[id].v[i]);
    }
  }
  G[id].color = BLACK;//探索済み
  G[id].fin = ++t;
}

void dfs(){
  int i,j;
//深さ優先探索の本質
  for(i=1; i<=n; i++){//初期設定でいったん全てのバーテックスを白にする
    G[i].color = WHITE;
  }
  t = 0;//探索ステップを0に初期化
  for(i=1; i<=n; i++){
    if(G[i].color == WHITE){
      visit(G[i].id);
    }
  }
}

int main() {
  int i, j;
  int id;

  //------各種入力処理------
  printf("要素の個数を入力(Input element count)");
  scanf("%d",&n);
  G = malloc(sizeof(Graph) * (n+1));
  getDef();
  for (i = 1; i <= n; i++) {
```

```
    printf("要素の ID を入力してください。Please input element ID\n");
    scanf("%d",&id);
    G[id].id = id;//ID 番目に ID を直接入れることで順番を守る
    printf("[%d]の要素の子供の数を指定してください。Please decide [%d]'s childlen
count.\n",G[id].id, G[id].id);
    scanf("%d",&G[id].k);
    if(G[id].k != 0){
      G[id].v = (int*)malloc(sizeof(int) * (G[id].k+1));
      for(j = 1; j <= G[id].k; j++){
  printf("[%d]の子供の ID を入力してください。Please input [&d]'s childlen ID.\n",G[id].id,
G[id].id);
  printf("%d / %d\n",j, G[id].k);
        scanf("%d",&G[id].v[j]);
      }
    }
  }
  dfs();
  printer();

  free(G);
  free(Adj);
  return 0;
}
```

G[ID]: 1 Discover Time: 1 Finished Time: 8
G[ID]: 2 Discover Time: 2 Finished Time: 7
G[ID]: 3 Discover Time: 9 Finished Time: 12
G[ID]: 4 Discover Time: 4 Finished Time: 5
G[ID]: 5 Discover Time: 3 Finished Time: 6
G[ID]: 6 Discover Time: 10 Finished Time: 11