Solve the following problems. The number of points for each problem is shown next to the problem and in the table below. The outcomes corresponding to each question are also shown. Use only the space provided to solve each problem.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(a) / 10</td>
<td>2,3</td>
</tr>
<tr>
<td></td>
<td>(b.1) / 10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(b.2) / 10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(a) / 8</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(b) / 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(c) / 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(d) / 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(e) / 8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>/ 10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/ 10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/ 10</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>/ 100</td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 (30 points)
In the following program, a map is represented using two formats. The first format is a two-dimensional array map2d. map2d[i][j]=1 if nodes i and j are connected by a road; otherwise, map2d[i][j]=0. Note that each road connects the two nodes at its endpoints in both directions. The second format is represented by the array map.

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

#define MAXN 5
char map2d[MAXN][MAXN];
struct element
{
    int node;
    struct element *next;
} *map[MAXN];

int main(void)
{
    void fillmap2d(),printmap2d(),translate(),printmap();

    fillmap2d();
    printmap2d();
    translate();
    printmap();
    printf("\n");
}

void fillmap2d()
{
    int i,j;

    for(i=0;i<MAXN;i++)
        for(j=i+1;j<MAXN;j++)
        {
            map2d[i][j]=rand()%2;
            map2d[j][i]=map2d[i][j];
        }
}
```
void printmap2d()
{
    int i,j;

    printf("n\nmap (format 1):");
    for(i=0;i<MAXN;i++)
    {
        printf("n%d:",i);
        for(j=0;j<MAXN;j++)
            if(map2d[i][j]) printf(" %d",j);
    }
}

void translate()
{
    int i,j;
    struct element *p;

    for(i=0;i<MAXN;i++) map[i]=NULL;
    for(i=0;i<MAXN;i++)
        for(j=i+1;j<MAXN;j++)
            if(map2d[i][j])
            {
                p=malloc(sizeof(struct element));
                assert(p!=NULL);
                p->node=j;
                p->next=map[i];
                map[i]=p;

                p=malloc(sizeof(struct element));
                assert(p!=NULL);
                p->node=i;
                p->next=map[j];
                map[j]=p;
            }
}
void printmap()
{
    int i,j;
    struct element *p;

    printf("n\nmap (format 2):\n");
    for(i=0;i<MAXN;i++)
    {
        printf("n%"i",i);
        for(p=map[i];p!=NULL;p=p->next) printf(" %d",p->node);
    }
}

(a) Suppose that the function fillmap2d() fills map2d so that it contains the following values. Show what the array map would contain after calling the function translate(). Include all the information that is relevant to the representation of the map exactly as it is stored.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
(b) Suppose that the map is changed so that roads now have a direction. map2d[i][j]=1 now implies that there is a road leading from node i to node j (it does not mean that there is a road from j to i).

(b.1) On the copy of the program given above, mark how the function translate() needs to be changed to accommodate the new definition of direction. Make as few changes as possible. There is no need to consider the other functions.

(b.2) Suppose that a modified version of the function fillmap2d() fills map2d so that it contains the following values. Show what the array map should contain after calling the modified function translate(). Include all the information that is relevant to the representation of the map exactly as it is stored.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Problem 2 (40 points)
For every one of the following programs, specify the files that will change after running the program. In addition, specify the contents of these files after running the program. File contents before running every one of the programs are as follows.
fileA:
    abcede1234
fileB:
    7w8x9
fileC:
    hello

(a)

#include <stdio.h>

int main(void)
{
    char ch;
    FILE *pf1,*pf3;

    pf1=fopen("fileA","r");
    pf3=fopen("fileC","w");

    while(fscanf(pf1,"%c",&ch)==1)
        fprintf(pf3,"%c",ch);

    fclose(pf1);
    fclose(pf3);
}

Name(s) of the file(s) that change:

Contents of these files:
(b)

```c
#include <stdio.h>

int main(void)
{
    char ch1,ch2,ch3,ch4;
    FILE *pf1,*pf3;

    pf1=fopen("fileA","r");
    pf3=fopen("fileC","w");

    while(fscanf(pf1,"%c%c%c%c",&ch1,&ch2,&ch3,&ch4)==4)
        fprintf(pf3,"%c%c%c%c",ch1,ch2,ch3,ch4);

    fclose(pf1);
    fclose(pf3);
}
```

Name(s) of the file(s) that change:

Contents of these files:
#include <stdio.h>

int main(void)
{
    char ch1,ch2;
    FILE *pf1,*pf2,*pf3;

    pf1=fopen("fileA","r");
    pf2=fopen("fileB","r");
    pf3=fopen("fileC","w");

    while(fscanf(pf1,"%c",&ch1)+fscanf(pf2,"%c",&ch2)==2)
        fprintf(pf3,"%c%c",ch1,ch2);

    fclose(pf1);
    fclose(pf2);
    fclose(pf3);
}

Name(s) of the file(s) that change:

Contents of these files:
#include <stdio.h>

int main(void)
{
    char ch;
    FILE *pf2,*pf3;

    pf2=fopen("fileB","r");
    pf3=fopen("fileC","w");

    fscanf(pf2,"%*[0123456789]");
    while(!feof(pf2))
    {
        fscanf(pf2,"%c",&ch);
        fprintf(pf3,"%c",ch);
        fscanf(pf2,"%*[0123456789]");
    }

    fclose(pf2);
    fclose(pf3);
}

Name(s) of the file(s) that change:

Contents of these files:
(e)

#include <stdio.h>

int main(void)
{
    char ch;
    FILE *pf1,*pf3;

    pf1=fopen("fileA","rb");
    fseek(pf1,-1,2);
    pf3=fopen("fileC","w");

    while(ftell(pf1)>0)
    {
        fscanf(pf1,"%c",&ch);
        fseek(pf1,-2,1);
        fprintf(pf3,"%c",ch);
    }
    fclose(pf1);
    fclose(pf3);
}

Name(s) of the file(s) that change:

Contents of these files:
Problem 3 (10 points)
The following program is compiled and run using the command
   a.out file1 file2
What will the program print, and what else will it do?

   #include <stdio.h>
   #include <stdlib.h>
   #define MAXSTR 100

   int main(int argc, char *argv[])
   {
      char str[MAXSTR];
      FILE *fp;

      sprintf(str,"cp %s %s",argv[1],argv[2]);
      printf("\n\n\n",str);
      system(str);
   }

The program will print:

In addition the program will:
Problem 4 (10 points)
What will the following program print?
Note that the ASCII code of the digit '1' is one more than the ASCII code of the digit '0', the ASCII code of the digit '2' is one more than the ASCII code of the digit '1', and so on.

```
#include <stdio.h>

#define MAXSTR 5

#define INITSTR(s) s[0]=0;

#define INSERTSTR(s,c) \ 
  for(p=s; *p;p++); \ 
  for(p>=s;p--) *(p+1)=*p; \ 
  s[0]=c;

int main(void)
{
  int i;
  char str[MAXSTR+1],*p;

  INITSTR(str);
  for(i=0;i<MAXSTR;i++)
  {
    INSERTSTR(str,'0'+i);
    printf("str=\%s\n",str);
  }
}
```

The program will print:
Problem 5 (10 points)

Complete the following program so that it would fill the array \(a\) with \(N\) random numbers in the range \([R1,R2]\). The program should also fill the array \(b\) so that \(b[i]\) will be the number of appearances of the number \(i\) in \(a\), for \(R1 <= i <= R2\).

Note: The program should work correctly if the values of the constants \(R1\), \(R2\) and \(N\) are changed. You can assume that \(R1\), \(R2\) and \(N\) will remain positive and \(R2 >= R1\).

```c
#include <stdio.h>
#include <stdlib.h>
#define R1 1
#define R2 50
#define N 20

int main(void)
{
    int i,r,a[_________],b[_________];

    for(i=_________;i<=_________;i++) b[i]=0;

    for(i=0;i<N;i++)
    {
        r=rand()______________________________;

        a[_________]__________;

        b[_________]__________;
    }

    printf("n\'na: ");
    for(i=0;i<N;i++) printf(" %d",a[i]);

    printf("n\'nb: ");
    for(i=R1;i<=R2;i++) printf(" %d",b[i]);
    printf("n");
}
```