ECE264 Advanced C Programming
TTh 7:30-8:20am
Exam 1

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 problem 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>/ 25</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>/ 25</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>/ 50</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td>/ 100</td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 (25 points)
Specify what the following program will print.

```c
#include <stdio.h>

#define N 4

int main(void)
{
    int i;
    int a[N]={2,1,3};
    void arrayfunc(int *, int);

    for(i=0;i<N;i++)
    {
        arrayfunc(a+i,N-i);
        printf("%d\n",a[i]);
    }
}

void arrayfunc(int *a, int n)
{
    int i;
    void swap(int *, int *);

    for(i=0;i<n;i++)
        if(a[i]<a[0]) swap(a,a+i);
}

void swap(int *pa, int *pb)
{
    int tmp;

    tmp=*pa;
    *pa=*pb;
    *pb=tmp;
}
```

The program will print:

```
0
1
2
3
```
Problem 2 (25 points)
Specify what the following program will print when it is run as shown below.

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

#define N 10
#define M 5

int main(int argc, char *argv[]) {
    int n,m,i,end;
    char t[M][N];
    void printrow(char [], int);
    int fillrow(char [][N], int, int);

    n=atoi(argv[1]);
    assert(n<=N);

    for(i=0;i<n;i++) t[0][i]=0;
    for(end=0,m=0;!end&&!(m<M);m++)
    {
        printrow(t[m],n);
        end=fillrow(t,m,n);
    }
    printf("\n\n");
}

int fillrow(char tab[][N], int m, int n) {
    int i;

    for(i=0;i<n;i++) tab[m+1][i]=tab[m][i];
    for(i=n-1;i>=0;!(tab[m+1][i]==1);i--) tab[m+1][i]=0;

    if(i==1) return(1);

    tab[m+1][i]=1;
    return(0);
}
```
void printrow(char row[], int n)
{
    int i;

    printf("\n");
    for(i=0;i<n;i++) printf("%d",row[i]);
}

(a) The program is run using the command
    a.out 1
The program will print:

    o
    |

(b) The program is run using the command
    a.out 2
The program will print:

    o o
    o |
    ( o
    | |

(c) The program is run using the command
    a.out 3
The program will print:

    o o o
    o o |
    o | o
    o | |
    | o o

(d) The program is run using the command
    a.out 4
The program will print:

    o o o o
    o o o |
    o o | o
    o o | |
    | o o
Problem 3 (50 points)

This problem has three parts.

(a) In the following program, mark statements that will give compilation errors. Mark every such statement with an arrow pointing to the statement, and provide a brief explanation next to the arrow. You should not mark any statements that will not give compilation errors.

(b) Delete from the program every part that depends on the statements you marked in (a). If you marked a statement within a function, delete the function and the function call. Also delete any other part of the program that depends on this function. Cross out deleted parts clearly. Do not cross out any parts that are not affected by the compilation errors.

The result of (a) and (b) should be a program that will compile and run correctly, and has as much of the functionality of the original program as possible. This should be achieved only by deleting statements. You cannot rewrite or add any statements.

```c
#include <stdio.h>

#define N 100

struct location
{
    unsigned x:3, y:3, z:3;
};

int main(void)
{
    struct location locations[N], smax;
    int nlocations, imax;
    int readlocations1(struct location []);
    int readlocations2(struct location []);
    int findmax1(struct location [], int);
    struct location findmax2(struct location [], int);
    void printlocation1(struct location [], int, struct location);
    void printlocation2(struct location [], int, int);

    nlocations = readlocations1(locations);
    nlocations = readlocations2(locations);
    imax = findmax1(locations, nlocations);
    smax = findmax2(locations, nlocations);

    printlocation1(locations, nlocations, smax);
    printlocation2(locations, nlocations, imax);
}
```
int readlocations1(struct location locations[])
{
    int nlocations,i;
    scanf("%d",&nlocations);
    for(i=0;i<nlocations;i++)
        scanf("%d%f%d",&locations[i].x,&locations[i].y,&locations[i].z);
    return(nlocations);
}

int readlocations2(struct location locations[])
{
    int nlocations,i,x,y,z;
    scanf("%d",&nlocations);
    for(i=0;i<nlocations;i++)
    {
        scanf("%d%f%d",&x,&y,&z);
        locations+i->x=x;
        locations+i->y=y;
        locations+i->z=z;
    }
    return(nlocations);
}

int findmax1(struct location locations[], int nlocations)
{
    int max,i;
    max=0;
    for(i=0;i<nlocations;i++)
        if(locations[i].x>locations[max]) max=i;
    return(max);
}
struct location findmax2(struct location locations[], int nlocations)
{
    struct location max;
    int i;

    max.x=0;
    max.y=0;
    max.z=0;

    for(i=0;i<nlocations;i++)
        if((locations[i].x>max.x) ||
           ((locations[i].x==max.x)&&(locations[i].y>max.y)) ||
           ((locations[i].x==max.x)&&(locations[i].y==max.y) &&
            (locations[i].z>max.z)))
            max=locations[i];

    return(max);
}

void printlocation1(struct location locations[], int nlocations,  
            struct location loc)
{
    int i;

    for(i=0;i<nlocations&&((locations[i].x!=loc.x))
            (locations[i].y!=loc.y)&&(locations[i].z!=loc.z);i++)
        printf("n[%d] %d %d\n",i,loc.x,loc.y,loc.z);
}

void printlocation2(struct location locations[], int nlocations, int iloc)
{
    printf("n[%d] %d %d\n",iloc,locations[iloc].x,locations[iloc].y, 
            locations[iloc].z);
}
(c) For the program that passes compilation, specify what the program will print for the input given below.

4
1 2 3
4 1 5
3 1 4
4 1 1

The program will print:

\[ [1] \ 4 \ 1 \ 5 \]