

CSE 110  
Final Exam  
Summer Session 1, 2001

name:	
e-mail:	

question	total points	points awarded
1	10	
2	15	
3	10	
4	10	
5	10	
6	15	
7	10	
8	10	
9	10	
Total	100	

1. (10 points) A data structure is needed to hold the information for all of the musicians who were at one time members of some band. For each musician, the following data must be stored:
- name - a character array of maximum size MAXSIZE (some defined constant)
  - instrument - one of GUITAR, DRUMS, BASS, NONE. This should be done using an enumerated type.
  - singer - TRUE or FALSE - a boolean
  - beginning year - an integer representing the first year the musician played with the band.
  - ending year - an integer representing the last year the musician played with the band.

Write:

- a `typedef` for an enumerated type `InstrumentT` for the instruments
- a `typedef` for a `struct` type named `MusicianT` that will include the above information.
- a declaration for a data structure that will hold this information for each of 10 different musicians.

*Answer:*

```
#include <stdio.h>
#include "simpio.h"

#define MAXSIZE 10

typedef enum {GUITAR,DRUMS,BASS,NONE} InstrumentT;
typedef struct {
    char name[MAXSIZE];
    InstrumentT instrument;
    bool singer;
    int beginning;
    int ending;
} MusicianT;

void main() {
    MusicianT band[10];
}
```

2. (15 points) In the following code `arr` is declared to be a 2-dimensional array of size `MAXSIZExMAXSIZE`. Suppose it's initialized somehow.

Write a function `GetMax` that will return the biggest integer in the array. You can assume that all the integers in the array are nonnegative. You can refer to `MAXSIZE` in the function `GetMax`.

```
#define MAXSIZE 5

void main() {

    int arr[MAXSIZE][MAXSIZE];
    int max;

    /* array gets initialized somehow */

    max=GetMax(arr);
    printf("The maximum value in the array is %d\n",max);
}
```

*Answer::*

```
int GetMax(int a[][MAXSIZE]) {

    int i,j;
    int max=0;

    for (i=0; i<MAXSIZE; i++) {
        for (j=0; j<MAXSIZE; j++) {
            if (a[i][j]>max) {
                max=a[i][j];
            }
        }
    }
    return max;
}
```

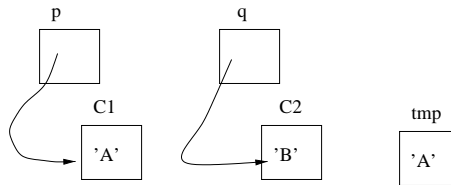
3. (10 points) Consider the following declarations and statements:

```
char c1='A', c2='B', tmp;  
char *p;  
char *q;
```

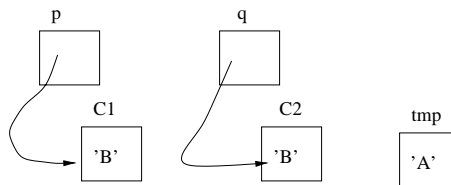
```
p=&c1;  
q=&c2;
```

Draw a diagram of what is in memory after each of the statements has been executed. The first statement comes after the code above, the second statement comes after the first statement has been executed, and the third statement comes after the second statement has been executed.

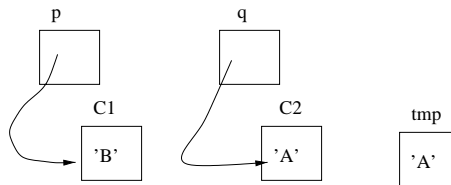
```
tmp=*p;
```



```
*p=*q;
```



```
*q=tmp;
```



4. (10 points) Consider the following code. The function call `RandomInteger(1,6)` returns a random number between 1 and 6. The intent of the programmer is that `val1` and `val2` should be set to two random values. Instead, the `printf` statement always prints out `val=0` and `val2=0`. Modify the code so that `val1` and `val2` will be set to random values and the `printf` statement prints out those random values. You cannot call `RandomInteger` within the `main()` function.

```
void GetTwoRandomValues(int val1, int val2)
{
    val1=RandomInteger(1,6);
    val2=RandomInteger(1,6);
}

void main()
{
    int val1=0;
    int val2=0;

    GetTwoRandomValues(val1,val2);
    printf("val1=%d and val2=%d\n",val1,val2);
}
```

*Answer:*

```
void GetTwoRandomValues(int *val1, int *val2)
{
    *val1=RandomInteger(1,6);
    *val2=RandomInteger(1,6);
}

void main()
{
    int val1=0;
    int val2=0;

    GetTwoRandomValues(&val1,&val2);
    printf("val1=%d and val2=%d\n",val1,val2);
}
```

5. (10 points) What does the following program output?

```
#include <stdio.h>

typedef struct {
    char* name;
    int numBurgers;
} Burgermeister;

void transaction ( Burgermeister* ps, Burgermeister s, int adjustAmount ) {
    (ps->numBurgers)=(ps->numBurgers)+adjustAmount;
    (s.numBurgers)=(s.numBurgers)-adjustAmount;
}

void main() {
    Burgermeister b;

    b.name="Meister Burger";
    b.numBurgers=15;

    transaction(&b, b, 5);
    printf("Burgermeister %s has %d burgers.\n",
           b.name, b.numBurgers);
}
```

*Answer:*

```
%a.out
Burgermeister Meister Burger has 20 burgers.
%
```

6. (15 points) Write prototypes for the functions `func1`, `func2`, `func3` so that the following code compiles. You don't need to write the functions - just the prototypes.

```
void main()
{
    int *p,*q;
    int i,j;

    i=func1(p,q);
    *p=func2(&i,*q);
    q=func3(i,*p);
}
```

*Answer:*

- `func1`: *Answer:*

```
int func1(int *, int *);
```

- `func2`:

*Answer:*

```
int func2(int *, int);
```

- `func3`:

*Answer:*

```
int *func3(int, int);
```

7. (10 points) Increase the efficiency of the monster puppet factory by converting the following variable declarations from a parallel array implementation to an implementation using a single array of one structure type.

```
#define NUMPUPPETS 25
#define MAXNAMELENGTH 100

char name[NUMPUPPETS][MAXNAMELENGTH];
int numberofeyes[NUMPUPPETS];
bool bleedsRealBlood[NUMPUPPETS];
```

*Answer:*

```
#include <stdio.h>
#include "simpio.h"

#define NUMPUPPETS 25
#define MAXNAMELENGTH 100

typedef struct {
    char name[MAXNAMELENGTH];
    int numberofEyes;
    bool bleedsReal;
} PuppetT;

void main() {

    PuppetT array[NUMPUPPETS];

}
```

8. (10 points) Given the following declarations and initializations:

```
typedef struct {
    double avg;
    int *hws;
} StudentT;

void main() {

    StudentT *student;
    student = (StudentT *)malloc(sizeof(StudentT));
    student->hws = (int *)malloc(2*sizeof(int));
    Init(student);

}
```

Assume that the function `Init()` initializes all storage pointed to by `student` appropriately. What *type* do each of the following expressions have?

- (a) (2pts) `student->avg`  
*Answer:double*
- (b) (2pts) `student->hws`  
*Answer:int \* or int[]*
- (c) (2pts) `student->hws[1]`  
*Answer:int*
- (d) (2pts) `*student`  
*Answer:StudentT*
- (e) (2pts) `student`  
*Answer:StudentT \**

9. (10 points) The code on the next page is intended to do the following:

- input up to MAXSIZE(5) integers.
- If any integer is less than the previous entered integer, than the integer just entered is treated as a sentinel (that is, it is not included in the list of integers, and no more integers are entered.)
- The main function prints out the number of integers entered (the effective size of the array), and what those integers are.

For example, this page shows some desired output from the working version of a program (which is not the code listed on the next page!)

The code listed on the next page has some errors in it. Some may be compile time, and some may be runtime. That doesn't mean that there are necessarily both compile and runtime errors.

What are the errors in the code, and how would you fix them?

Sample output:

```
% a.out
Enter an integer:10
Enter an integer:9
1 numbers were entered
numbers[0]=10
```

```
%a.out
Enter an integer:1
Enter an integer:2
Enter an integer:3
Enter an integer:4
Enter an integer:5
5 numbers were entered
numbers[0]=1
numbers[1]=2
numbers[2]=3
numbers[3]=4
numbers[4]=5
```

```
% a.out
Enter an integer:6
Enter an integer:7
Enter an integer:8
Enter an integer:9
Enter an integer:0
4 numbers were entered
numbers[0]=6
numbers[1]=7
numbers[2]=8
numbers[3]=9
%
```

```

#include <stdio.h>
#include "simpio.h"

#define MAXSIZE 5

int GetNumbers(int numbers[]) {
    int numEntered=1;
    int val;

    printf("Enter an integer:");
    numbers[0]=GetInteger();

    while (numEntered<MAXSIZE) {
        printf("Enter an integer:");
        val=GetInteger();
        if (val<numbers[numEntered-1]) {
            return (numEntered);
        } else {
            numbers[numEntered]=val;
            numEntered++;
        }
    }
    return MAXSIZE;
}

void main()
{
    int numEntered;
    int numbers[MAXSIZE];
    int i;

    numEntered=GetNumbers(numbers []);
    printf("%d numbers were entered\n",numEntered);
    for (i=0; i<=numEntered; i++) {
        printf("numbers [%d]=%d\n",i,numbers [i]);
    }
}

```

*Answer:*

- In call to `GetNumbers` in `main()`, can't have `[]` after `numbers`
- In `for` in `main`, should be `<NumEntered`, not `<=Entered`