cs246
11/11
Lecture

1. extern int yyyy
    1. usage
    2. Example
2. “struct TAG {};” vs “typedef struct {} NAME;” vs “typedef struct TAG {} NAME;”
    “struct TAG” came from era in which C did not have typedef
    IMO mostly still useful for recursive structures (linked lists, etc)

3. sed -e, like grep -e

4. Getting hostname of machine

Q15 Write a recursive function to find the position of the last instance of any character in a set of characters in the the target
The function should have the signature int lastpos(char* target, char* lastset). The function can be destructive of target.
for example, lastpos(“this is a test”, “as”) would return 12.
ANSWER

```
#include<stdio.h>
#include<string.h>

int lastposc(char* tgt, char* c, int cc)
{
    if (cc==0) return -1;
    int v = lastposc(tgt, c, cc-1);
    int ll = strlen(tgt);
    int vv=-1;
    for (int l=(ll-1); l>=0; l--)
    {
     if (tgt[l]==c[cc])
     {
         vv=l;
         break;
     }
    }
    return (v>vv) ? v : vv;
}

int lastpos(char* tgt, char* chars)
{
    return lastposc(tgt, chars, strlen(chars));
}

int main()
{
    printf("%d\n", lastpos("this is a test", "as"));
29   }
```

Q18 Short answer. What are the advantages / disadvantages of storing structs directly in arrays as opposed to arrays of struct pointers.
ANSWER
Advantages of direct storage: speed of access, flexibility, simplicity, less space required
Disadvantages: Structs are passed by value for use of structs can be very slow, resizing arrays of structs can be very expensive

Q10.
Consider the .h file named stdhelper.h:
```
#ifndef  STDHELPER
#define STDHELPER
typedef struct helpstruct {
     int help;
} Helpstruct;
#endif
```
Why is the #ifndef .. #define considered good practice in .h files?
ANSWER:
Multiple calls to typedef will result in compile time errors. This methods ensures that typdef is only executed once.

Q27. The ackerman-peter function is defined as follows:

A(m,n) = n+1 if m==0
= A(m-1, 1) if m>0 and n==0
= A(m-1, A(m, n-1)) otherwise

Write a C implementation of this function.
Will does this function terminate? Explain.
Show the beginning of a trace for the calls of this function when started with A(4,2)? That is, show all recursive calls and recursive returns
ANSWER:
discussed in class

Q20 Consider the following program (numbers at left are line numbers — not part of the program)
```
 1   #include<stdio.h>
 2   typedef struct testres {
 3       char fst[20];
 4       char lst[20];
 5       int score;
 6       char grade;
 7   } testres;
 8   testres stu2;
 9   int main()
10   {
```

```
11      testres stu1;
12      testres res[30];
13      printf("%s", stu1);
14      printf("%d %c", stu1.score, stu2.fst);
15      stu2=stu1;
16      if (stu2.grade == stu1.grade) printf("Cheating???");
17      if (stu2==stu1)  printf("Same structure”);
18  }
```
for each line itentify it as “possibly valid” or “defineitly invalid”. If invalid, suggest a fix.

ANSWER:

```
line 17 is definitely bad, it is rejected by the compiler
line 13 is definitely odd and will not print anything reasonable
line 14 will print things, but nothing that is meaningful
line 15 works, copying stu1 over stu2
```

Q30 Write a function to set the bits in a byte. The function should take an array of 7 integers; all either 0 or 1. It should return a single byte where the bits of the byte are identical to, and in the same order as, those of the 7 integers

ANSWER:

```
#include<stdio.h>

int main()
{
    int a[] = { 0, 0, 1, 0, 0, 1, 0};
    int b = 1;
    int rr = 0;
    for (int i=6; i>=0; i--)
    {
    rr += a[i] * b;
    b=b*2;
    }
    printf("%d\n", rr);
    char c = rr;
    printf("%d\n", c);
}
```

Q1Suppose these two files:

debug.h

```
#ifdef DEBUG
#define PRINTD(n) printf("Value %d\n", n)
#else
#define PRINTD(N)
#endif
```

debug.c

```
 1   #include <stdio.h>
 2   #include "deb.h"
 3   #define DEBUG
 4   int main() {
 5       int i=1, j=2, k=3;
 6   #ifdef DEBUG
 7       printf("Debug defined\n");
 8   #else
 9       printf("DEBUG not defined\n");
10   #endif
11       PRINTD(k);
12       return 0;
13   }
```
[
What is output of program?
ANSWER:

    Debug Defined

What is output if line 3 of debug.c is removed?
ANSWER:

    DEBUG not defined

What is the output if line 3 of debug.c is moved to between lines 1 and 2
ANSWER:

    Debug defined
    Value 3

What is output if #define DEBUG removed and compiled with “-DDEBUG”

    Debug defined
    Value 3

Q2 Suppose that a program consists of 3 source files: main.c, f1,c and f2.c and 2 header files: f1.h and f2.h. All three source files contain f1.h but only f2.c contains f2.h. Write a makefile
ANSWER:

```
ff: f1.0 f2.o main.o
    gcc –o ff main.o f1.0 f2.o

f1.o: f1.c f1.h
    gcc –c main.c

main.o: main.c f1.h
    gcc –c f1.c

f2.o: f1.c f1.h f2.h
    gcc –c f2.c
```

Q3
Give the following file
expan.c

```
#define ASDF(x,y) (fabs(x-y) - fabs(y-x))

int main()
{
    if (ASDF(a+b, c) > ASDF(b+c, a))
     printf("HELLO");
}
```

```
what will the main function look like after executing gcc -E expan.c
ANSWER:
int main()
{
    if ((fabs(a+b-c) - fabs(c-a+b)) > (fabs(b+c-a) - fabs(a-b+c)))
 printf("HELLO");
}
```

Q5. Let a be an array of 100 integers. Write a call to qsort that sorts only the last 50 elements in a. Write the comparator function for qsort as well.
ANSWER

```
#include<stdio.h>
#include<stdlib.h>
int c(const void * p, const void * q)
{
    int pi = *((int *)p);
    int qi = *((int *)q);
    return pi-qi;
}

int main()
{
    int a[100];
    srand(40);
    for (int i=0;i<100; i++) a[i] = rand() % 100;
    qsort((a+50), 50, sizeof(int), c);
    for (int i=0; i<100; i++)
     printf("%d  %d\n", i, a[i]);
}
```

```
Note that the call to qsort is slightly different than the one I gave
in class on 11/11
```

Q6. Write a comparator function and a call to qsort that will sort aaa in the code below. Imagine that qsort is called at line 12 (replacing the comment).

```
1   #include<stdio.h>
2   #include<stdlib.h>
3   #define SIZ 30
4   typedef struct {
5       int a;
6   } A;
```

```
int main()
{
    srand(SIZ);
    A aaa[SIZ];
    for (int i=0; i<SIZ; i++) aaa[i].a=rand()%SIZ;
    //qsort to be written
    for (int i=0; i<SIZ; i++) printf("%d  %d\n", i, aaa[i].a);
}
```

ANSWER:

```
#include<stdio.h>
#include<stdlib.h>
#define SIZ 30
typedef struct {
    int a;
} A;

int c(const void * p, const void * q)
{
    A pa = *((A *)p);
    A qa = *((A *)q);
    return pa.a - qa.a;
}

int main()
{
    srand(SIZ);
    A aaa[SIZ];
    for (int i=0; i<SIZ; i++) aaa[i].a=rand()%SIZ;
    qsort(aaa, SIZ, sizeof(A), c);
    for (int i=0; i<SIZ; i++) printf("%d  %d\n", i, aaa[i].a);
}
```

Q7. Given the structure defined below.

```
typedef struct LL {
    int a;
    struct LL * ap;
} LL;
```

write 2 functions

LL* fillLL(int N)

that creates N linked instances of LL. Each instance should contain an pseudorandom integer in the range 0..10 inclusive.

int countn(LL* al, int n)

that counts the number of times the value n appears in LL

ANSWER:

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

typedef struct LL {
    int a;
    struct LL * ap;
} LL;

LL* head;

void fillLL(int n)
{
    head = malloc(sizeof(LL));
    head->a = rand() % 11;
    LL* p = head;
    for (int i=0; i<(n-1); i++)
    {
    LL* nxt = malloc(sizeof(LL));
    nxt->a = rand() % 11;
    p->ap = nxt;
    p = nxt;
    }
}

int countn(LL* h, int n)
{
    int rtn = 0;
    while(h!=NULL)
    {
    if (h->a==n) rtn++;
    h=h->ap;
    }
    return rtn;
}

int main()
{
    srand(50);
    fillLL(50);
    printf("%d \n", countn(head, 7));
}
```

Q16 Write a recursive function with the signature
    void bitter(int tgt)
that prints the bits of an integer. for instance,
    bitter(8) would print 1000
    bitter(10) would print 1010
The only library function you can use is putchar

ANSWER:

```
#include<stdio.h>

int main()
{
    int v = 129;
    int b = 1;
    while (b<v) b=b*2;
    b=b/2;
    while (b>=1)
    {
     if (v>=b)
     {
         v=v-b;
         putchar('1');
     }
     else
         putchar('0');
     b=b/2;
    }
    putchar('\n');
}
```

Q28.
The periodic table contains elements. Each element has a name of at most 20 characters and number of protons and a most common number of neutrons. Elements also have 1 or more isotopes. This data is stored in a file with lines like:

carbon 6 6 12 13 14
nitrogen 7 7 14 15
oxygen 8 8 14 15 16 17 18 19 20

Write a structure for efficiently (in terms of space) holding this information
Write a function for reading from a file with lines as above

ANSWER:

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

typedef struct {
    char* name;
    int protons;
    int neutrons;
    int* isotopes;
} element;

element* readElementLine(char* line)
{
    element* e = malloc(sizeof(element));
    int isos[100];
    int isocount = 0;
    char* dta;
    char* dd = strtok(line, " ");
    e->name = malloc((strlen(dd)+1)*sizeof(char));
    strcpy(dd, e->name);
    dd=strtok(NULL, " ");
    e->protons = atoi(dd);
    dd=strtok(NULL, " ");
    e->neutrons = atoi(dd);
    while (NULL != (dd=strtok(NULL, " ")))
    {
    isos[isocount++]=atoi(dd);
    }
    e->isotopes = malloc(isocount*sizeof(int));
    for (int i=0; i<isocount; i++)
    e->isotopes[i]=isos[i];
    return e;
}
int main()
{
}
```