YORK UNIVERISTY
DEPARTMRENT OF COMPUTER SCIENCE AND ENGINEERING

Midterm

CSE 2031 Software Tools

June 8, 2010

SOLUTIONS

Family Name: _____
Given Name: _____
CS account: _____
Student Id: _____

Instructions:

1. The exam has 10 questions and 8 pages (including this one)
2. No aids permitted
3. Answer each question in the space provided. If you need more space write on the backs of pages
4. Examination time is 80 minutes
5. Do *not* use *red* ink.
6. Write legibly. Unreadable answers do not count.

Question	Max	Mark
1	5	
2	8	
3	3	
4	3	
5	2	
6	2	
7	6	
8	2	
9	4	
10	5	
Total	40	

1. (5 points) How many bytes of memory are allocated for each of the following variables? Assume that a character takes 1 byte, integer takes 4 bytes, and floating point number takes 4 bytes.

```
[1]     float a[] = {4.7};
```

Answer: 4 (one float)

```
[2]     char a[] = "4.7";
```

Answer: 1*4=4 (four characters including '\0')

```
[3]     char animals[][10] = {"lion", "elephant", "tiger", "cat"};
```

Answer: 1*10*4=40 (four arrays containing 10 characters)

```
[4]     union {int x; float y} number;
```

Answer: 4 (size of largest type = float)

```
[5]     int c[10] = {1,2,3};
```

Answer: 10*4=40 (ten integers)

---

2. (8 points) Consider the following definitions in ANSI-C program:

```
int i = 3, k[]={2,4,6,8,10,12}, *x=&i, *y=k;
double d=1.5;
struct point {   int x;
                 int y;
                 char *name;
} pt[] = {{200,40,"begin"},{300,100,"end"}}, *pp = pt;
```

For each of the following expressions, fill in the type of the expression and its value. The type you give must be a valid ANSI-C type (e.g. char for a character, int* for a pointer to an integer etc.) If you think the expression is invalid, write *invalid* for the type and leave value blank. Each expression is independent. No explanation is necessary.

Expression	Type	Value
[1] ++pp->x	int	300
[2] pt[1].x*i/5	int	180
[3] k[i++]	int	8
[4] i+k[++i]+d	double	14.5
[5] pt[i--].y+50	Invalid	pt[3] points to unallocated memory
[6] strcmp((*pp).name,"begin")	int	0
[7] *pp.name+2	Invalid	. has higher priority than *
[8] (*pp).name+2	char*	Pointer to 'g'

3. (3 points) Indicate whether each of the following statements is TRUE or FALSE

1. The glass-box testing method chooses test cases based upon requirement specification.

   Answer: TRUE / FALSE

2. It is a good idea to ask someone who did not design the program or implement the code to create test cases for the program.

   Answer: TRUE / FALSE

3. In regression testing we should run only new test cases to test the fix for a bug.

   Answer: TRUE / FALSE

---

4. (3 points) Declarations

1. Declare a variable `api` to be an array of 100 pointers to `int`.

Answer:

```
int *api[100]
```

2. Declare a variable `nf` to be a pointer to a function which takes two `ints` and returns a pointer to `float`.

Answer:

```
float *(* nf)(int, int)
```

3. Define a new type `SF` for function pointers to be used to point to function which takes a `char` pointer as argument and returns a pointer to void.

Answer:

```
typedef void* (*SF)(char *)
```

5. (2 points) What impact does the keyword `static` have on the variable?

Answer: A variable declared as static in a function is initialised once, and retains its value between function calls. The default initial value of an uninitialized static variable is zero. If a function or global variable is declared static, it can only be accessed in that file.

---

6. (2 points) What impact does the keyword `extern` have on the variable?

Answer: Declaring a variable as extern will result in your program not reserving any memory for the variable in the scope that it was declared. An extern is something that is defined externally to the current module.

---

7. (6 points) Each of the following short progarmes contains a programming error (not necessarily a syntax error). State clearly what the error is. Provide line number when specifying the error.

1. (3 points)

```
/* 1 */ #include <stdlib.h>
/* 2 */ #include <stdio.h>

/* 3 */ int main(){

/* 4 */       typedef struct {
/* 5 */            int start;
/* 6 */            int end;
/* 7 */            char letter;
/* 8 */       }note;

/* 9 */       note *p;
/* 10 */      p = (note *)malloc(2*sizeof(int)+sizeof(char));
/* 11 */      if(p==NULL) exit(1);
/* 12 */      p->start=80; p->end=100; p->letter='A';
/* 13 */      return 0;
 /* 14 */     }
```

Answer: line 10:sizeof(note) may be different than sum of sizeof of each memeber

Line 11 and line 13: exit without freeing memory (bonus point)

2. (3 points)

```
/* 1 */    #include <stdlib.h>
/* 2 */    #include <stdio.h>

/* 3 */    int *findMax(int *a, int *b){
/* 4 */          int max;
/* 5 */          if(*a > *b) max=*a;
/* 6 */          else max = *b;
/* 7 */          Return &max;
/* 8 */    }

/* 9 */    int main(){
/* 10 */         int x=7, y=15, max;
/* 11 */         max = findMax(&x, &y);
/* 12 */         return 0;
/* 13 */   }
```

Answer: line 7: pointer to local variable returned

Line 10 and 11: max is int, findMax returns pointer

Line 7: return should be lowercase

---

8. (2 points) Consider the following program:

```
#include <stdio.h>
int rec(int x, int y){
      static int count = 0;
      if(x==0)
            return count;
      count++;
      if(x>y)
            rec(x-y,y);
      else
            rec(x,y-x);
      return count;
}

main(){
      int i=10, j=2, n;

      n=rec(i,j);
}
```

What is the value of `n` at the end of the execution of this program?

Answer: infinite loop

9. (4 points) Write a function `replace` that has following signature:

```
int replace( char *s, char a, char b);
```

The function replaces all the occurances of character a in string s by character b, and returns the number of characters it replaced. Do not use functions from string.h. For example:

```
char x[] = "test";
int n;
n = replace(x, 't','z');
```

Should set `x` to "`zesz`" and `n` to `2`;

```
/* Replaces all the occurrence of a in the string pointed by s with b*/

int replace(char *s, char a, char b)

{/* YOU WRITE THE REST ...*/

      int count = 0;   /* will keep no of characters replaced */
      While(*s!=NULL){ /* until null is reached do*/
            if(*s==a){ /* if got char equal to a then replace and
                          increase counter*/
                  *s=b;
                  count++;
            }
            s++;       /* move pointer */
      }
      return count;    /* return counter */
}
```

10. (5 points) Write a function `reverse` that has following signature:

```
void reverse( char *s);
```

The function reverses string s passed as argument. For example:

```
char x[] = "test";
int n;
n = reverse(x);
```

Should set `x` to `"tset"`.

Use only a constant amount of extra space, not dependent on the length of the passed string. Use standard library if necessary. Assume that all required headers are included.

```
/* Reveres the string pointed by s*/

void reverse(char *s)

{/* YOU WRITE THE REST ...*/
```

```
char *start, *end, c; /* pointers to firs and last, and temp */
int n;                /* length of string */
n=strlen(s);
if(n>1)               /* if 0 or 1 nothing to do*/
{
      for(start=s, end=s+n-1; start<end; start++, end--)
      {/* swapping last and first until end passes start*/
            c = *start;
            *start = *end;
            *end = c;
      }
}
```

(This page is intentionally blank to provide extra space)