CSC 111 Fall 2011
Midterm 2

7½ / TWO

- Turn in your completed midterm **at the front of the class; UVic ID check;** leave through the front door on your left.
- **Do not leave before 10:45 am.**

1. Consider the following syntactically correct C declarations and assignments. **[8]**

```
int x;
int y;
int *p;
int *q;
int** t;
x = 44;   p = &x;
q = p;    y = 19;
t = &q;
```

What are the values of the following expressions?

(&x == p)	0 False not defined.	TRUE
(p == &y)	0 false not defined	FALSE
(*q == 17)	0 False. not defined	false
(**t== *p)	0 False. type dash.	true
(y == x)	0 False.	False
(*q == 44)	0 False.	True
(x == p)	0 False type dash.	typedash
(&x == *t)	0 False.	~~typedash~~ TRUE

&x = q
&x = *t

8

2. Consider the following declarations: **[6]**

```
#include <string.h>
typedef struct {
        char first[20];
        char last[20];
        float salary;
} Person;
Person student;
```

Initialize variable **student** with your **first** and **last** name as well as your dream **salary**. **Hint:** Use function defined in <string.h>.

```
Person student;
strcpy(student.first, "Teresa");
strcpy(student.last, "Cutler");
student.salary = 80000;
```

```
strcopy(student.first, "Teresa");
strcopy(student.last, "Cutler");
student.salary = 60000;
```

6

3. Consider the following declarations: **[4]**

```
float* fp;
void* vp;
```

How do you assign **vp** to **fp** properly using a cast or type conversion?

```
typedef void* float*.
float*fp = *vp;
fp = (float**)vp;
```

```
fp = (float*)vp;
```

0

6

4. Consider the following syntactically correct C declarations and assignments. **[6]**

```
int a;
int *b;
int **c;
a = 17;
b = &a;
c = &b;
```

c=&b, b=&a

Using one **printf()** statement output the **address** of variable **b** and the **address** of variable a.

```
printf("%p,%p\n", c, b);

printf("Address of b = %p Address of a = %p\n", c, b);
```

5. In the C programming language, how do you refer to a file when you read, write or close a file? **[4]**

- (✓) FILE*
- ( ) fopen
- ( ) printf
- ( ) fgetc

6. A complex number consists of two parts: a real (**re**) and imaginary (**im**) part. Which of the following code fragments correctly defines a structure type "Complex"? **[4]**

typedef struct {
re
im
} complex

- ( ) structure { double re; double im; } Complex;
- (✓) typedef struct {double re, im; } Complex;
- ( ) typedef struct Complex {double re, im }
- ( ) struct Complex (double re, im;);

7. Consider the following syntactically correct C program called 'reflection.c'. What happens when you execute this program? **[8]**

```
#include<stdio.h>
#include<stdlib.h>
#define MAX (300)
#define FNAME  "reflection.c"
int main(void) {
        char line[MAX];
        FILE *ifp;
        ifp = fopen(FNAME, "r");
        if (ifp == NULL) {
                printf("Input file %s not found\n", FNAME);
                exit(EXIT_FAILURE);
        } /* if */
        while(!feof(ifp)) {
                if (fgets(line, MAX, ifp)) printf("%s", line);
        } /* while */
    fclose(ifp);
    return EXIT_SUCCESS;
} /*main*/
```

- ◯ This program will create a new file.
- ✓ This program will output the program text of this program.
- ◯ This program will count the number of lines in file 'reflection.c'
- ◯ This program will copy file 'reflection.c' to a new file 'reflection.java'.

8. What is the output of the following syntactically correct C program? **[6]**

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    int k = 3;
    for (k=5;  k < 12; k++) {
        printf("%d ",  k%7);
    } /* for */
    printf("\n");
    return EXIT_SUCCESS;
} /* main */
```

Output: 5 6 0 1 2 3 4

9. What is the console output of the following syntactically correct C program? **[8]**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define  OUTPUT   (" Once upon a time there was a polar\n")
int main(void){
        char str[] = OUTPUT;
        FILE *ifp;
        FILE *ofp;

        ofp = fopen("csc111.txt", "w");
        fputs("Programming is really cool!\n", ofp);
        fclose(ofp);
        ifp = fopen("csc111.txt", "r");
        while(fgets(str, strlen(str), ifp) != NULL)  printf("%s", str);
        printf("My favorite course is CSC 111!\n");
        fclose(ifp);
        return EXIT_SUCCESS;
} /* main */
```

- ○ Once upon a time there was a polar bear
  My favorite course is CSC 111!
- ✓ My favorite course is CSC 111!
- ○ Programming is really cool!
  My favorite course is CSC 111!
- ○ Programming is really cool!

10. Which of the following is true? **[4]**

- ○ Each component of a struct is assigned the same chunk of storage space
- ○ The syntax for structs is basically the same as for arrays
- ○ Each component of a struct is of same type
- ✓ Structs are also called records

11. Insert a syntactically correct **print()** statement into the following C code—where the box is—to output the **second to last** character of the string **s.str** using the length of the string. **[8]**

length-2.

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

#define MAX_SIZE  (30)
typedef struct {
   int length;
   char str[MAX_SIZE ];
}  StringDesc;

int main(void) {
   StringDesc s;
   strcpy(s.str, "Melanie Amaro of X FACTOR");
   s.length = strlen(s.str);
```

printf("%c", s.str[length-2]);

```
   return EXIT_SUCCESS;
} /* main */
```

printf("s.str[s.length-2] = %c" \n", s.str[s.length-2]);

12. Write a syntactically correct C function to swap the values of two integer variables. **[8]**

```
#include <stdlib.h>
#include <stdio.h>
void swap(int*a, int*b){
   int tmp;
   tmp = *b;
   *b = *a;
   *a = tmp;
return; }

int main void(){
   int a = 2;
   int b = 1;
   printf("%d, %d", a, b);
   swap(&a, &b);
   printf("%d, %d", a, b);
return; }
```

```
void swap(int* x, int* y){
   int tmp;
   tmp = *x;
   *x = *y;
   *y = tmp;
}
```

```
Void swap(int* x, int* y)
{ int tmp;
   tmp = *x;
   *x = *y;
   *y = tmp;
}
```

13. What is the effect of the following initialization? **[8]**

```
#include <stdio.h>
#include <stdlib.h>
#define vSize (4)
typedef int Item;
typedef int Index;
typedef Item Vector[vSize];
void initVector(Vector V, Index size, Item z) {
   Index k;
   for (k=0; k<size; k++) V[k] = (Item)(k*z);
} /* initVector */
void printVector(const Vector V, Index size) {
   Index k;
   for (k=0; k<size; k++) printf("%d  ", V[k]);
} /* printVector */
int main(void) {
   Vector Vec;
   initVector(Vec, vSize, 5);
   printVector(Vec, vSize);
   initVector(Vec, vSize, 7);
   printVector(Vec, vSize);
   printf("\n");
   return EXIT_SUCCESS;
} /*main*/
```

- ○ 1 5 11 16 0 4 7 10
- ○ 0 7 13 17 2 5 8 11
- ✓ 0 5 10 15 0 7 14 21
- ○ 0 7 14 21 0 5 10 15

14. How do you sort the variables **a** and **b** in decreasing order by calling routine rigi? **[4]**

```
void rigi(int* x, int* y) {
   if (*x < *y)  int tmp = *x; *x = *y; *y = tmp;
} /* rigi */
```

- ○ int a = 3, b = 17; rigi (3, 17);
- ○ int a = 3, b = 17; rigi(a, b);
- ✓ int a = 3, b = 17; rigi(&a, &b);
- ○ int a = 17, b = 3; rigi();

15. What is the output of the following syntactically correct C program? **[8]**

```
#include <stdio.h>
#include <stdlib.h>
#define VSIZE  (4)
typedef float Vector[VSIZE];
void func1(Vector a, int len) {
        int k;    float first = a[0];
        for (k=0; k<len-1; k++)  a[k] = a[k+1];
        a[len-1] = first;
} /*func1*/
void func2(Vector a, int len) {
        int k;
        for (k=0; k<len; k++)  printf("%.1f ", a[k]);
        printf("\n");
} /*func2*/
int main(void) {
        Vector vec;
        vec[0] = 1.1;    vec[1] = 5.5;    vec[2] = 4.4;    vec[3] = 3.3;
        func1(vec, VSIZE);        func1(vec, VSIZE);        func2(vec, VSIZE);
        return EXIT_SUCCESS;
} /* main */
```

- ○ 4.4 3.3 5.5 4.4
- ○ 5.5 4.4 3.3 1.1
- ○ 1.1 5.5 4.4 3.3
- ✓ 4.4 3.3 1.1 5.5

16. The following syntactically correct C code initializes an array buffer of size 16 with the char '#'. Which for loop below replaces all the array elements with an odd index with the character '@' so that the array contains the following string: "#@#@#@#@#@#@#@#@" **[6]**

```
#define len 16
char buffer[len];
int k;
for (k=0; k<len; k++) buffer[k] = '#';
```

- ○ for (k=0; k<len; k++) buffer[k] = '@';
- ○ for (k=0; k<len; k++) buffer[k] = '#@';
- ✓ for (k=0; k<len; k++) if (k%2 == 0) buffer[k] = '@';
- ○ for (k=0; k<len; k++) if (k%2 != 0) buffer[k] = '@';