Name:____________ Solutions_____________________

Student ID: ____________________________

**HONOR CODE:** At the end of the examination, before submitting, please sign:

*In recognition of the integrity of the engineering profession and the UCSC Honor Code, I certify that I have neither given nor received un-permitted aid during this examination.*

Signature:________________________________________

This exam is closed book, no notes. You get no credit for anything that is not understood by the graders. **Show all work and if you want partial credit please be neat!**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100</strong></td>
<td></td>
</tr>
</tbody>
</table>
1. (10 pts) *Arithmetic and Logic Operators.*
   A) (8 pts) A and B are both 8-bit values. A = 0x3E and B=0x5A. The result should also be at most 8-bits.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &amp;&amp; B</td>
<td>1</td>
</tr>
<tr>
<td>A &amp; B</td>
<td>0001 1010 or 0x1A</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>A ^ B</td>
<td>0110 0100 or 0x64</td>
</tr>
<tr>
<td>A &lt;&lt; 3</td>
<td>1111 0000 or 0xF0</td>
</tr>
<tr>
<td>A &gt;&gt; 2</td>
<td>0000 1111 or 0x0F</td>
</tr>
<tr>
<td>A++ + ++B</td>
<td>1001 1001 or 0x99</td>
</tr>
</tbody>
</table>

B) (2 pts) What is the result of the follow statement if I=4, J=5, D= -5, E=2? How should it be written to be more clear?

\[ X = I \times J + D \times E / 2 \]

\[ X = (I \times J) + ((D \times E)/2) \]

\[ X = 15 \]

2. (5 pts) *Variables.*
   A) (3 pts) Create a symbol table for this code fragment. Assume all data types have the same size.

```c
char bob;
int joe;
main () {
    int fred;
    float george;
    ...
}
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>bob</td>
<td>char</td>
<td>0</td>
<td>Global</td>
</tr>
<tr>
<td>joe</td>
<td>int</td>
<td>1</td>
<td>Global</td>
</tr>
<tr>
<td>fred</td>
<td>int</td>
<td>0</td>
<td>main</td>
</tr>
<tr>
<td>george</td>
<td>float</td>
<td>-1</td>
<td>main</td>
</tr>
</tbody>
</table>

B) (2 pts) Declare two arrays. One of integers that have the values: 1, 2, 3, 4, 5, 6, 7, 8. The second of characters that spell “Programming is fun!”.

```c
int myintarray[]= {1,2,3,4,5,6,7,8};
char mychararray[]="Programming is fun!";
```
3. (7 pts) C Function Programming.
Create a .h and a .c file that declares and defines a function called “total_cost()”. The function should be documented appropriately. Pass a price, tax rate, and discount percent to the function. The function should calculate and return the total amount due. The discount should be applied before the tax is calculated.

Contents of the “.h” file. Name of file _____myfun.h_____________

```
// Tax and Discount are in percentage
float total_cost (float price, float tax, float discount);
```

Contents of the “.c” file. Name of file _____myfun.c_____________

```
// Inputs: price of an item
//         tax rate as a percentage
//         discount as a percentage

float total_cost (float price, float tax, float discount) {
    float total;

    // calculate price after discount
    total = price - price * (discount/100);
    // add the tax in
    total += price * (tax/100);
    // return result
    return (total);
}
```

Convert this code to use a “do-while” loop, not a “for” loop?

```
main () {
    int i;
    for (i=25; i >=0; i--)
        printf("%c", 'z'-i);
    printf("\n");
}
```

```
main () {
    int i=25;

    do {
        printf("%c","z'-i);
        i--;
    } while (i>=0)

    print("\n");
}
```
5. (5 pts) **Pointers.**

Describe what lines 5 through 10 each do.

```
1:   int i;
2:   char name [5];
3:   int *iptr;
4:   char *cptr;
5:   iptr = &i;
6:   cptr = name;
7:   i = 4;
8:   name[2] = 'b';
9:   *iptr = *iptr + 1;
10:  cptr = cptr + 1;
```

- Store the address of i into the memory location associated with ptr
- Store the address of name[0] into the memory location associated with cptr
- Store the value 4 into the memory location associated with i
- Store the character “b” into the array “name” at element 3
- Read the contents of memory at the address stored in ptr
- Increment the address stored at location cptr by 1
- Store the result into memory at the address stored in ptr

6. (5 pts) **UNIX Command Line.**

A) (3 pts) Give the command line arguments for creating a directory, then for changing locations to be inside of this new directory, and finally to get a listing of the files inside.

```
> mkdir bob
> cd bob
> ls
```

B) (2 pts) Give the command line argument for how you would search through all C files looking for the string “Fred”.

```
> grep “Fred” *.c   or   grep Fred *.c
```

7. (4 pts) **Printing in C.**

The following lines of C code appear in a program. What will be the output of each `printf` statement?

```c
#define LETTER 'l'
#define ZERO 0
#define NUMBER 123

printf("%c", 'a');
printf("x%x", 23);
printf("$%d.%c%d\n", NUMBER, LETTER, ZERO);
```

**Solution:**

```
a
x17
$123.10
```
8. **(13 pts) Short Questions.**

A) **(3 pts)** Compare and contrast the execution process of an interpreter versus the execution process of a compiled binary. What implications does interpretation have on performance, if any?

**Solution:**
With compilation, the program is first transformed from a high level language into machine code. The machine code version can be directly executed by the hardware. With interpretation however, the program is processed by an interpreter (in other words, the program is input data for the interpreter), which executes it line-by-line, section-by-section, method-by-method. Since the underlying computing hardware executes the interpreter, which then indirectly executes the user program, the extra layer adds execution overhead.

B) **(1 pts)** What purpose does the semicolon serve in C?

**Solution:**
The semicolon in C terminates a statement.

C) **(2 pts)** Rewrite the following code to use the conditional operator (hint: ?) rather than using an if-else.

```c
if (a==b) x = a; else x = c;
```

**Solution:**

```
x = (a==b) ? a : c;
```

D) **(4 pts)** Describe the behavior of the following 2 code snippets for the case when x equals 0 and when x equals 1?

```c
if (x = 0)
    printf (“x equals 0\n”);
else
    printf (“x does not equal 0\n”);
```

**Solution:**
The expression (x = 0) always assigns 0 to the variable x. Since this expression as a whole will likewise evaluate to 0, "x does not equal 0" will be printed in all cases.

```c
int x;
int y;
...
switch (x) {
    case 0:
        y = 3;
    case 1:
        y = 4;
        break;
    default:
        y = 5;
        break;
}
```

**Solution:**
When x equals 0:
The first case is true. All lines of code after the case and before a break statement (or the end of the switch block) are executed. So, “y = 3”; is executed, and then “y = 4”; is executed before the break is encountered.

When x equals 1:
The second case is true, so “y = 4” is executed before the break is encountered.

E) **(3 pts)** In regards to an activation record, what is:

- The purpose of the dynamic link?

**Solution:**
When a function is called, the address of the caller's activation record on the runtime stack is referred to as the dynamic link. The dynamic link enables the caller’s frame pointer to be restored when the callee completes.

- The return address?

**Solution:**
The return address is the address of the next instruction in the caller to be executed when the callee returns control. The return address enables us to resume execution in the caller once the callee completes.

- The return value?

**Solution:**
The return value is the output (actually, one of the outputs) of a function.
9. **(4 pts) Scope in C.**
   For the following code, give the values that are printed out by each `printf` statement.

   ```c
   int t; /* This variable is global */
   {
      int t = 2;
      printf("%d\n", t);
      {
         printf("%d\n", t);
         t = 3;
      }
      printf("%d\n", t);
   }
   {
      printf("%d\n", t);
      {
      }
      printf("%d\n", t);
   }
   Solution:
   2
   2
   3
   0 (or anything)
   ```

10. **(5 pts) C programming.**
    Write a program that reads an integer from the keyboard and displays a 1 if it is divisible by 3 or a 0 otherwise.

    ```c
    Solution:
    #include <stdio.h>
    int main()
    {
      int input; char output;
      scanf("%d", &input);
      /* There are many ways to accomplish this. Here is one that does not use conditional expressions. */
      output = ((input % 3) == 0) + '0';
      printf("%c", output);
    }
    ```

11. **(4 pts) C programming.**
    What is the output of the following code?

    ```c
    int main () { int x = 1; int y = 2;
    swap (x, y); printf ("x = %d y = %d\n", x, y);
    }
    void swap (int y, int x) {
    int temp;
    temp = x; x = y; y = temp;
    }
    Solution:
    The output of the program is “x = 1 y = 2”. Even though the function Swap() swaps the values of the two arguments, these values are only swapped in the activation record for Swap. From the perspective of function main(), the values of x and y remain unchanged.
12. (12 pts) Pointers.

A) Declare a structure that contains: An integer called ID. Character arrays called first and last, each of 10 characters. A pointer to the structure called next.

B) Write a function called delete_ID() that given a pointer to the head of a linked list of structures of the type you just created and an ID, will search through the linked list and remove the node with that ID and print the First and Last names out. If the ID is not present it will print out “No such ID present”. Example of the function call:
delete_ID (myhead, myID);

```c
struct mystuct {
    int ID;                    /* ID of person */
    char first[10];           /* max 10 characters */
    char last[10];            /* max 10 characters */
    struct mystuct *next;    /* pointer to struct mystuct */
};

struct mystuct* delete_ID (struct mystuct *myhead), int myID) {
        struct mystuct *savehead, *prev;

        /* keep a copy of the start of the head */
        savehead = myhead;

        /* Check if head has the ID */
        if (myhead->ID == myID) {
                printf ("First = %s\n", myhead->first);
                printf ("Last = %s\n", myhead->last);
                prev = myhead;
                myhead = myhead->next;
                free (prev);
                return (myhead);
        }
        /* Move the pointer to the next node since not the first node */
        myhead = myhead->next;

        prev = myhead;
        while (myhead != NULL) {
                if (myhead->ID == ID) {
                        printf ("First = %s\n", myhead->first);
                        printf ("Last = %s\n", myhead->last);
                        prev->next = myhead->next;
                        free (myhead);
                        return (savehead);
                }
                else {
                        prev = myhead;
                        myhead = myhead->next;
                }
        }
        printf ("No such ID\n");
        return(savehead);
```
13. (5 pts) GDB Fun.

Use the gdb source-level debugger to monitor the execution of the following code. Use vi/emacs to enter the code. For what values of parameters does the function Mystery return a zero?

```c
#include <stdio.h>

int main()
{
    int i; /*iterations variable */
    int j; /*iterations variable */
    int k; /*iterations variable */
    int sum=0; /*running sum of Mystery */

    for (i=25; i > 0; i--)
    {
        for (j=1; j < i; j++)
        {
            for (k=j; k < 50; k++)
            {
                sum = sum + Mystery (i, j, k);
            }
        }
    }

    int Mystery(int a, int b, int c)
    {
        static max = 1000;
        int out;

        out = 3*a*a + 7*a - 5*b*b + 4*b + 5*c;

        return out;
    }
```


A) Create a file that contains just character text and spaces (see the example below).
B) Write a program that takes an integer (the key) and your file name from the command line. Your program should shift the characters a-z and A-Z in the file by the key, and account for “roll-around” at the end of the alphabet. The key can be from 0 to 25.
D) Write the results to a new file called encrypted.
C) Submit this code and a Makefile to compile it to the locker “midterm”.

Example of execution of the program:  
```
> mycrypt 2 mytextfile
mytextfile: This is the end of z
encrypted: Vjku ku vjg gpf qh b
```

Example:
```
Solutions:
24, 21, 45
17, 16, 46
16, 15, 37
```
#include <stdio.h>
#include <stdlib.h>

char shift_character (int shift, char input_char);

int main(int argc, char *argv[]){
    FILE *read_fp, *write_fp;
    char in_char, out_char;
    int shift;

    // open file to read
    if((read_fp=fopen(argv[2],"r")) == NULL) {
        printf("Cannot open file to read.\n");
        exit(1);
    }
    // open file to write
    if((write_fp=fopen("encrypted","w")) == NULL) {
        printf("Cannot open file to write.\n");
        exit(1);
    }
    // convert the shift to an integer
    shift = atoi(argv[1]);

    while (fscanf(read_fp,"%c",&in_char) != EOF) {
        out_char = shift_character (shift, in_char);
        fprintf(write_fp,"%c",out_char);
    }

    fclose(read_fp);
    fclose(write_fp);
    return 0;
}

// Take a character and an integer and shift the
// character by the integer, key from 1 to 25
char shift_character (int shift, char input_char) {
    unsigned char output_char;

    // Check if lower case
    if (input_char >= 'a' && input_char <= 'z') {
        output_char = input_char + shift;
        if (output_char > 'z')
            output_char = output_char - 26;
    }
    else if (input_char >= 'A' && input_char <= 'Z') {
        output_char = input_char + shift;
        if (output_char > 'Z')
            output_char = output_char - 26;
    }
    else
        output_char = input_char;

    return output_char;
}