# ECE 15 Fall 16

## Midterm Solutions

This is a closed-book exam: no notes, books, calculators, cellphones, or friends are allowed.

In problems 4–6, you can assume that the user’s input is correct.

If you need more space, please use the back of this page.

You have 80 minutes.

**Good luck!**

---

Name: ____________________________

PID: ____________________________

---

<table>
<thead>
<tr>
<th>Points</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ________ /20</td>
<td></td>
</tr>
<tr>
<td>2. ________ /7</td>
<td></td>
</tr>
<tr>
<td>3. ________ /7</td>
<td></td>
</tr>
<tr>
<td>4. ________ /22</td>
<td></td>
</tr>
<tr>
<td>5. ________ /22</td>
<td></td>
</tr>
<tr>
<td>6. ________ /22</td>
<td></td>
</tr>
<tr>
<td>= ________ /100</td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 (20 points)

For each of the following programs, indicate whether or not it has errors. If there are errors, mark all of them (there may be more than one) on the program and briefly describe the problems. If there are no errors, write the program’s output.

(1) #include <stdio.h>
    int main() {
        int i = 0;
        for(i = 37; i < 11; i+2) {
            i = i - 30;
            printf("%d ", i);
        }
        printf("%d", i);
        return 0;
    }

Output: 37 __ __ __ __ __ __ __

(2) #include <stdio.h>
    int main() {
        int a = 4;
        int b = 15;
        int c = 32;
        if(b%++a)
            printf("%d ", c++);
        else
            printf("%d ", c--);
        printf("%d", c);
        return 0;
    }

Output: 32 31 __ __ __ __ __ __
(3) #include <stdio.h>  
#define LIMIT = 5;  
#define START = 0;  
int main() {  
    int i, j;  
    for(i = START; i < LIMIT; i++) {  
        j = i%3;  
        while(j < START)  
            printf("%lf ", ++j);  
    }  
    return 0;  
}  

(4) #include <stdio.h>  
int main() {  
    int i;  
    int arr[7] = {4,2,7,1,9,3,6};  
    for (i = 1; i < 7; ++i) {  
        arr[i-1]+=arr[i];  
        arr[i]=arr[i-1]-arr[i];  
        arr[i-1]-=arr[i];  
    }  
    for (i = 0; i < 7; ++i)  
        printf("%d ", arr[i]);  
    return 0;  
}
**Problem 2 (7 points)**

Write the output of the following program.

```c
#include <stdio.h>
int main(){
    int i=0,j=0,x=2,y=1;
    while(x>y??++i;++j){
        if(x<y)
            x*=y;
        else
            y*=x;
        if(x==y)
            j-=(x*y-1);
        printf("x=%d, y=%d, i=%d, j=%d\n", x, y, i, j);
    }
    return 0;
}
```

```
x=2, y=2, i=1, j=-3
x=2, y=4, i=1, j=-2
x=8, y=4, i=1, j=-1
x=8, y=32, i=2, j=-1
```
Problem 3 (7 points)
Rewrite the switch-case part of the following program using if-else statements.

```c
#include <stdio.h>
int main() {
    int num;
    scanf("%d", &num);
    switch (num) {
        case 2: printf("num is even.\n");
        case 7: printf("num is prime.\n"); break;
        case 3: printf("num is odd.\n");
        case 5: printf("num is prime.\n");
        case 1: printf("num is odd.\n"); break;
        default:
            (num%2) ? printf("num is odd.\n") : printf("num is even.\n");
    }
    return 0;
}
```

```c
#include <stdio.h>
int main() {
    int num;
    scanf("%d", &num);
    if (num==2){
        printf("num is even.\n");
        printf("num is prime.\n");
    }
    else if (num==7)
        printf("num is prime.\n");
    else if (num==3){
        printf("num is odd.\n");
        printf("num is prime.\n");
        printf("num is odd.\n");
    }
    else if (num==5){
        printf("num is prime.\n");
        printf("num is odd.\n");
    }
    else if (num==1)
        printf("num is odd.\n");
    else
        (num%2) ? printf("num is odd.\n") : printf("num is even.\n");
    return 0;
}
```
Problem 4 (22 points)

Write a program that asks the user for a single character and then for a sequence of 0 or more characters. The program removes all occurrences of the single character from the sequence, and prints the remaining characters, switching the case of the letters from lower-case to upper-case and vice versa. Do not use the C string library.

Describe your rough approach in a few lines:

```c
#include <stdio.h>
int main() {
    char c, current;
    printf("Character: ");
    c = getchar();
    while(getchar()!='\n') {
        printf("Sequence: ");
        while (((current = getchar()) != '\n')) {
            if (current == c)
                continue;
            if (current>='A' && current<='Z')
                current+='a'-'A';
            else if (current>='a' && current<='z')
                current+='A'-'a';
            putchar(current);
        }
        putchar('\n');
    }
    return 0;
}
```
Problem 5 (22 points)
Recall that the Newton-Raphson algorithm approximates a root of a function $f(x)$ by starting with an initial guess $x_1$, and successively updating $x_i = x_{i-1} - f(x_{i-1})/f'(x_{i-1})$.
Write a program that uses the Newton-Raphson algorithm to approximate $\sqrt[3]{a+4}$, the cube root of $a + 4$, for a double $a$. The program asks the user for the double values of $a$ and $x_1$, and iterates till the first time that $|x_i - x_{i-1}| < 1E - 7$, namely $10^{-7}$, a value defined as a macro. The program then prints the value of $x_i$ to a 6-digit accuracy. Do not use the C math library.

Describe your rough approach in a few lines:

```c
#include <stdio.h>
#define TOL 1e-7
int main() {
    double a, x, xold, fx, dfx;
    int flag=0;
    printf("Number a: ");
    scanf("%lf", &a);
    printf("Initial guess: ");
    scanf("%lf", &x);
    while (flag == 0 || !(xold-x<TOL && x-xold<TOL)) {
        fx = x*x*x-a-4;
        dfx = 3*x*x;
        if (dfx == 0)
            break;
        xold = x;
        x = x - fx/dfx;
        if (flag==0)
            flag=1;
    }
    printf("Cube root of a+4: %.6f\n", xold);
    return 0;
}```
Problem 6 (22 points)

Write a program that identifies missing values in a range. The program prompts the user for a minimum integer \( \text{min} \), a maximum integer \( \text{max} \), and an integer number of values \( N \). It then asks the user for a list of \( N \) distinct integers between \( \text{min} \) and \( \text{max} \) (inclusive) and prints, in order, all the numbers in this range not entered by the user. You can assume that the user input will satisfy \( 0 \leq \text{min} \leq \text{max} \) and \( 1 \leq N \leq \text{max} - \text{min} + 1 \). If any of the list integers is less than \( \text{min} \), or greater than \( \text{max} \), or repeats an earlier entered integer, print “Invalid Input!”.

Describe your rough approach in a few lines:
#include <stdio.h>

int main() { 
    int min, max, range, N, i, a, flag = 0;
    printf("Min and Max: ");
    scanf("%d%d", &min, &max);
    range=max-min+1;
    int num[range];
    for(i=0; i<range; i++)
        num[i]=0;
    printf("Number of integers: ");
    scanf("%d", &N);
    printf("List: ");
    for(i=0; i<N; i++) {
        scanf("%d", &a);
        if((a<min)||(a>max))
            flag = 1;
        else if(num[a-min]==0)
            num[a-min] = 1;
        else
            flag = 1;
    }
    if(flag)
        printf("Invalid input! ");
    else {
        printf("Missing: ");
        for(i=0; i<range; i++)
            if(num[i] == 0)
                printf(" %d", min+i);
        printf("\n");
        return 0;
    }
}