Final Solutions

This is a closed-book exam: no notes, books, calculators, cellphones, or friends are allowed.
In problems 2–7, you can assume that the user’s input is correct.
User input is underlined.
If you need more space, please use the blank pages in the back.
You have 3 hours.

Good luck!

Name: ____________________________

PID: A__________________________

<table>
<thead>
<tr>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ________/20</td>
</tr>
<tr>
<td>2. ________/13</td>
</tr>
<tr>
<td>3. ________/13</td>
</tr>
<tr>
<td>4. ________/13</td>
</tr>
<tr>
<td>5. ________/13</td>
</tr>
<tr>
<td>6. ________/13</td>
</tr>
<tr>
<td>7. ________/15</td>
</tr>
</tbody>
</table>

= ________/100
Problem 1 (20 points)

For each of the following programs, indicate whether or not it has errors, including syntax-errors and infinite loops. 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. Hint: two programs contain errors.

(1) 
```c
#include <stdio.h>
int main() {
    int arr[][2] = {1,2,3,4};
    printf("%d,%d", arr[1][0], arr[0][1]);
    return 0;
}
```

- No Error

Output: __ 3 , 2 __ __ __ __ __

(2) 
```c
#include <stdio.h>
int main() {
    static char s[15] = "Tritons";
    int i = 0;
    char ch;
    ch = s[++i];
    printf("%c", ch);
    ch = s[i++];
    printf("%c", ch);
    ch = s[i++];
    ch += 1;
    printf("%c", ch);
    return 0;
}
```

- No Error

Output: __ r r j __ __ __ __

(3) 
```c
#include <stdio.h>
void myPrint();
int main() {
    myPrint();
    return 0;
}
void myPrint() {
    char c = ‘k’;
    if(c != ‘\n’)
        myPrint();
    printf("%c", c);
}
```

- No Error

Output: __ __ __ __ __ __ __ __
(4) #include <stdio.h>
int i = 0;
void f();
int main() {
    f();
    printf("%d", i);
    f();
    return 0;
}
void f() {
    static int z = 8;
    printf("%d", z);
    z++;
    i++;
}

X No Error  □ Error
Output: 8 1 9

(5) #include <stdio.h>
int myFunc(int);
int main() {
    int i = myFunc(10);
    printf("%d", --i);
    return 0;
}
int myFunc(int x) {
    int i;
    for (i = 1; i < x; i++) {
        if (i == 4)
            break;
    }
    return (i++);
}

X No Error  □ Error
Output: 3

(6) #include <stdio.h>
#define FUNC(x) x*x/(3+x)
int main() {
    float t = 6, a;
    a = FUNC(t+1);
    printf("%.1f", a);
    return 0;
}

X No Error  □ Error
Output: 1 2 • 1
(7) 
```c
#include <stdio.h>
int main() {
    int x = 3, y = 2;
    int *z = &x;
    printf("%d", y**z*x+z);
    return 0;
}
```

Output: 2 1

(8) 
```c
#include <stdio.h>
int main() {
    void f(char*);
    char arr[4] = {'a', 'b', 'c', 'd'};
    f(arr[0]);
    return 0;
}
void f(char *a) {
    a++;
    printf("%c", *a);
    a++;
    printf("%c", *a);
    return a;
}
```

Output: 

(9) 
```c
#include <stdio.h>
int main() {
    char *s;
    s = "%s";
    printf(s, "%s");
    return 0;
}
```

Output: % s

(10) 
```c
#include <stdio.h>
int main() {
    int arr[3] = {2, 3, 4};
    int *p;
    p = arr;
    printf("%d", *p);
    p = p+1;
    printf("%d", *p);
    return 0;
}
```

Output: 2 3
Problem 2 (13 points)

Write a program that takes as command line arguments three positive integers X, Y, and Z. The program calculates the smallest and largest of the three elements \( \text{MIN} = \min\{X, Y, Z\} \) and \( \text{MAX} = \max\{X, Y, Z\} \), and prints their product \( \text{MAX} \times \text{MIN} \).

```
#include <stdio.h>

int main(int argc, char* argv[]) {
    int X, Y, Z, MAX, MIN;
    X = atoi(argv[1]);
    Y = atoi(argv[2]);
    Z = atoi(argv[3]);
    MAX = X>Y ? X : Y;
    MAX = MAX>Z ? MAX : Z;
    MIN = X<Y ? X : Y;
    MIN = MIN<Z ? MIN : Z;
    printf("MAX * MIN = %d\n", MAX*MIN);
}
```

Hint: Recall that main takes two arguments, int argc and char *argv[]. The latter is an array of strings, hence each of its elements is a string. You may want to use the functions scanf or atoi (we expect you to remember how to use them).

Describe your rough approach in a few lines:
Problem 3 (13 points)

Write a program that asks the user for a positive integer $n$ and evaluates $f(n)$ for a function $f$ defined as

$$f(n) = \begin{cases} 
3 & n = 1, \\
5 & n = 2, \\
2 \cdot f(n-1) - f(n-2) & n > 2.
\end{cases}$$

Your program should include the following recursive function.
- int myFunc(int n); that takes the integer $n$ and returns $f(n)$.

```
#include <stdio.h>
int myFunc(int n) {
    int x;
    if (n==1)
        return 3;
    if (n==2)
        return 5;
    else
        return 2*myFunc(n-1) - myFunc(n-2);
}
```
Problem 4 (13 points)

Write a program that asks the user of a positive integer, multiplies the digits at the same distance from the left and right, namely, the first and the last digits, the second and the second-last ones, etc. and returns the sum of these products. For example, if the input is 840251, it prints “(8*1)+(4*5)+(0*2)”, namely 28. If the integer has an odd number of digits, then the center digit is not multiplied by itself. For example, for 14532 the program prints “(1*2)+(4*3)+5”, namely 19. You may assume that the input has at most 10 digits.

```c
#include <stdio.h>

int main() {
    int a[10];
    int number, n_digits, sum = 0;
    printf("Number: ");
    scanf("%d", &number);
    int x = number;
    int i = 0;
    while (x > 0) {
        n_digits +=1;
        a[i] = x%10;
        i++;
        x /= 10;
    }
    for (int j = 0; j<n_digits/2; j++) {
        sum += a[j]*a[n_digits-j-1];
    }
    if (n_digits % 2 == 1)
        sum += a[n_digits/2];
    printf("Sum: %d\n", sum);
    return 0;
}
```

Describe your rough approach in a few lines:
Problem 5 (13 points)

For security purposes, we want to encode words by breaking them into two parts after a given number of characters, reversing the characters in the first part, and then printing the second part followed by the reversed first part. For example, for the word “nocart” and breaking index 2, the two parts are “no” and “cart”, the first part reversed is “on”, and the encoded word is “carton”.

Write a program that asks the user of a word of length at most 20, the integer breaking index (starting at 1), and prints the encoded message.

```
#include <stdio.h>

int main() {
    char a[20], e[20];
    int b, length, i=0;
    printf("Word: ");
    while (1) {
        a[i] = getchar();
        if (a[i]=='\n')
            break;
        i++;
    }
    printf("Breaking index: ");
    scanf("%d", &b);
    length = i;
    for (int j = b; j < length; j++) {
        e[j-b]=a[j];
    }
    for (int j = 0; j < b; j++) {
        e[j+length-b]=a[b-j-1];
    }
    printf("Encoded word: ");
    for (int j=0; j< length; j++) {
        printf("%c", e[j]);
    }
    printf("\n");
    return 0;
}
```
Problem 6 (13 points)

Write a program that opens an input-file “input.txt” whose first line specifies the size of a square matrix. The subsequent lines consist of space-separated double elements of a square matrix, each row in a separate line. The program reads the matrix from the input-file and calculates the mean of all matrix elements, namely their sum divided by the number of elements. It then subtracts the mean from all the matrix elements and prints the resulting matrix with one-digit precision in an output file “output.txt”. For example in the example shown below the mean 6.2 is subtracted from each matrix element. Assume that the input file exists and has the right format.

```c
#include <stdio.h>

int main() {
    int x, dim;
    FILE *fin = fopen("input.txt", "r");
    FILE *fout = fopen("output.txt", "w");
    fscanf(fin,"%d",&dim);
    double ar[dim][dim];
    double mean, sum = 0;
    for (int i=0; i<dim ; i++) {
        for (int j=0;j<dim; j++) {
            fscanf(fin,"%lf",&ar[i][j]);
            sum = sum + ar[i][j];
        }
    }
    mean = sum / (dim * dim);
    for (int i=0;i<dim;i++) {
        for (int j=0; j<dim; j++) {
            fprintf(fout,"%.1lf",ar[i][j]-mean);
            if (j != dim-1)
                fprintf(fout," ");
        }
        fprintf(fout,"
");
    }
    fclose(fin);
    fclose(fout);
    return 0;
}
```

Describe your rough approach in a few lines:
Problem 7 (15 points)

The pattern of a string of characters indicates the order at which characters appear. The first character observed is marked as 1, the second as 2, etc. For example, the pattern of “cat” and “dog” is 123. The pattern of “ece” and “mom” is 121. The pattern of “aaa” is 111, while “cool” is 1223, of “banana” is 123232, and of “aardvark” is 11234125. Write a program that asks the user of a string of at most 20 lower-case characters and outputs its pattern. Your program should include the function below.

- **void pattern(char word[], int pat[], int length);** that takes character array *word[],* integer array *pat[],* and the length of the pattern as inputs, and stores the pattern in *pat[].*

In the **main** function of the program you should find the length of the pattern and call the function *pattern.*

```
(-)$ a.out
Word: egg
Pattern: 122
(-)$ a.out
Word: banana
Pattern: 123232
(-)$ a.out
Word: abracadabra
Pattern: 12314151231
(-)$
```

*Describe your rough approach in a few lines:*
#include <stdio.h>
void pattern(char [], int [], int length);
int main() {
    char a[20];
    int pat[20];
    pat[0] = 1;
    printf("Word: ");
    int i = 0;
    while (1) {
        a[i] = getchar();
        if (a[i] == '\n')
            break;
        i++;
    }
    int l = i;
    pattern(a, pat, l);
    printf("Pattern: ");
    for (int i = 0; i < l; i++) {
        printf("%d", pat[i]);
    }
    printf("\n");
    return 0;
}
void pattern(char a[], int pat[], int length) {
    int flag, temp, cur = 2;
    for (int i = 1; i < length; i++) {
        flag = 1;
        for (int j = 0; j < i; j++) {
            if (a[i] == a[j]) {
                flag = 0;
                temp = pat[j];
                break;
            }
        }
        if (flag == 1) {
            pat[i] = cur;
            cur++;
        } else
            pat[i] = temp;
    }
}