Eastern Mediterranean University
Department of Computer Engineering

CMPE 112 Midterm Exam
Fall Semester 2018-2019
November 27, 2018

Name, Surname :............................
Student No :...**SOLUTIONS**..............
Group No :.........

Instructors: *Assoc. Prof. Dr. Ekrem Varoğlu (Gr. 01), Prof. Dr. M. Güler (Gr. 02)*

Duration: 90 minutes

Instructions:

- There are **5** questions in this **9** page sheet. Please check !!!!!!
- Calculators are not allowed.
- GSM phones should be turned off.
- A table of operators for precedence and associativity is attached.
- Passing any material including rubbers, pencils etc. to anybody else is strictly prohibited in the exam.
- If an answer box is given in a question, you must give your answer (and nothing else) in the box !!!

Question	**Grade**
Q1(6 pts.)	
Q2(10 pts.)	
Q3(36 pts.)	
Q4(18 pts.)	
Q5(32 pts.)	
TOTAL: (out of 102)	

PRECEDENCE AND ASSOCIATIVITY

OPERATORS	ASSOCIATIVITY
( ) [ ] -> .	Left to right
! ++ -- + - * & (type) sizeof	Right to left (Unary)
* / %	Left to right
+ -	Left to right
< <= > >=	Left to right
== !=	Left to right
&&	Left to right
||	Left to right
?:	Right to left
= += -= *= /= %=	Right to left
,	Left to right

Q1) (6 pts)
Give the output of the following program:

```
#include <stdio.h>
main()
{
  int i, j, k;
  float x;
  i = 11; j = 4; k = i/j;
  printf(“%d\n”, k);

  x = i/j + 2.5;
  printf(“%4.2f\n”, x);

  k = i/j + 2.5;
  printf(“%d\n”, k);

  k = (float)i/j + 2.5;
  printf(“%d\n”, k);

  x = (float)i/j + 2.5;
  printf(“%4.2f\n”, x);

  x = (float)(i/j) + 2.5;
  printf(“%4.2f\n”, x);
}
```

```
2
4.50
4
5
5.25
4.50
```

Q2) (10 pts)
Find the values of the integer variables **i, j, k** after evaluating the given statements and the before values.

Before Values			*Statement*	*After Values*		
i	**j**	**k**		**i**	**j**	**k**
5	2	6	`i = j-- + k--;`	8	1	5
1	2	3	`i = j + (1 > k-j);`	2	2	3
1	2	3	`j *=!i != 5;`	1	2	3
1	2	3	`k = (i + j? k++ - 1 : j*2);`	1	2	2
1	3	7	`i = i % ( i < j );`	0	3	7
1	3	7	`i = (i < j)% j;`	1	3	7
1	5	7	`i=--i && --j || --k;`	1	5	6
1	3	1	`j=--i || --j && --k;`	0	0	0
1	3	7	`k = --i ? !i : !k;`	0	3	0
1	3	7	`k= (++i +4)/(--k - j--);`	2	2	2

Q3) (36 pts)
Give the outputs of the following programs or program segments:

a)

```
 #include <stdio.h>
  main()
  {
    int j, k;
    for(j=1,k=3; k <= 10; k += 2)
    {
       j += k;
       if(j>7) break;
    }
    printf("%d %d\n", j, k);
 }
```

```
9   5
```

----

b)

```
 #include <stdio.h>

 main()
 {
     int a=6,b=1,c=3;
     while(a<b<c)
     {
          switch(a<b<c)
          {
               default : ++a;
                      continue;
               case 1  : --a;--c;
               case 0  : ++b;
                      break;
          }
          printf("%d %d %d\n",a,b,c);
     }
 }
```

```
5 2 2
4 3 1
3 4 0
```

---

c)

```
  #include <stdio.h>
  main()
  {
     int a=0, b=6;
     do
     {
        printf("%d %d\n", a, b/2);
      }while(++a,  b -= a);
  }
```

```
0 3
1 2
2 1
```

d)

```
#include <stdio.h>
main()
{
 int x[5] = {0};
 int j, k;
 for(j=1; j < 5; j++)
  for(k=1; k < 5; k++) x[k] += j/k;
 for(j=0; j < 5; j++)
 printf("x[%d] = %d\n", j, x[j]);
}
```

```
x[0] = 0
x[1] = 10
x[2] = 4
x[3] = 2
x[4] = 1
```

----

e)

```
int a[3][3]={1,2,3,4,5,6};
int j=0, i;
while(a[j][j]) {
    for(i=0; i<3; i++) {
        a[2][j] += a[j][i];
    }
    j++;
}

for(i=0; i<3; i++) {
    printf("%d\n", a[2][i]);
}
```

```
6
15
0
```

----

f)

```
int f(int, int);
main()
{
   printf("%d  %d  %d\n", f(2,3), f(5,2), f(2,4));
}

int f(int x, int y)
{
 int k, z;
 z=1;
 for(k = 0; k < y; k++) z *= x;
 return z;
}
```

```
8  25  16
```

g)

```
 int k=3;
 main()
 {
   void fun(int);
   int y=6,  x=12;

   fun(x);
   printf("%d  %d  %d", k, y, x);
  }

  void fun(int a)
 {
   int y;

   k = a + 3;
   y = 8;
  a++;
}
```

15 6   12

Q4) (18 pts) The program below accepts a set of integer grades from the keyboard and stores them in an array named *grades*. The maximum number of grades to be entered is 50. The grades are counted as they are read and grade entry ends when a negative grade is entered (assume that the first grade entered is not negative, i.e. there is at least one non-negative grade). After all grades have been entered, the program should compute the average of the grades. Finally, the program must print all grades which are below the computed average. Fill in the missing parts of the program below accordingly.

```
#include <stdio.h>
#define MAXGRADE 50
main()
{
  int grades[MAXGRADE];
  int sum=0, count=0, i;

  /* count is used for counting number of grades entered*/
  float average;

  do
  {
     printf("Please enter the next grade\n");

     __scanf("%d", &grades[count])__; /*read the next grade */

     /*stop entering grades if the grade entered is less than 0 */
     /* otherwise add the grade to the sum */

     if (_grades[count]<0_____);

         ___break______________;
     else
     {
       sum+=_ grades[count]____;

       count++;
     }

   } while (_count<50_____);

     average=__(float)sum/count_____; /*calculate the average*/

      /* find and print the grades below the average */

      for( i=0; ___i<count__; i++)

        if(_grades[i]<average___)

          printf("%d is below the average\n", _grades[i]__);
   }
```

Q5) (32 pts) The program below finds the average of the elements of an array named *arr* using a function named *findAv* and prints the average in the main program.

a) Fill in the missing parts of the program below accordingly.

```
#include <stdio.h>

__float__ findAv(_int[]_,_int_);

int main()
{
  int grades[10]={90, 45, 18, 29, 100, 58, 47, 69, 82, 15};
  float average;

  average=findAv(_grades_ , 10);

  printf("average of array elements=%5.2f\n",average);

  return 0;

}

 __float__ findAv (_int grades[]_ , size)

{

 int i;

 float sum=0.0;

 for (i=__0_; _i<size_; i++)

   sum+=_grades[i]_;

 return (__sum/size__);

}
```

b) Fill in the missing blanks below for the function definition of a function named *shiftLeft* which will be called from the main program given in section a) in the previous page after the call to the *findAv* function. This function will receive the array *arr* and shift its elements by one position to the left. (i.e. the elements of the array after shifting will be: {45, 18, 29, 100, 58, 47, 69, 82, 15, 90}). The shifted array will be displayed in the main.

```
____void__shiftLeft (_int arr[]___ , size)

{

 int i,temp;

temp=arr[0]; /* shift the array elements one to the left here */

for (i=_0__; __i<9__; i++)

  __arr[i]=arr[i+1]__;

 arr[9]=__temp___;

return;

}
```