120 minutes **METU CEng 230 FINAL EXAM** **12.1.2008 A**

There are 17 questions for a total of 100 points.
Questions 1-12 are multiple choice (5 points each); 13–17 are short answers (8 points each).
Wrong answers do NOT cost you.
Partial credit up to 2 points may be granted in Questions 13–17.
Please write the answer that you think is most appropriate.
Assume a variable is properly declared if declaration is not given.

Name: **Student ID**: **Grade**: + =

1. (5 points) Assume arrays `x` and `y` are of same size, say `k`. What are the contents of arrays `x` and `y`, after the following function is called as `h(x,y,k);` ?

```
int h(int a[], int b[], int n)
{int i;
   for(i=0; i<n; i++)
     a[i] = a[i]+b[n-i-1];
   return (0);
}
```

   (a) They are unchanged.
   (b) They are both initialized to 0.
   (c) Array `x` is initialized to 0.
   (d) Array `x` contains addition of old contents of `x` to the corresponding element in reverse of `y`.

2. (5 points) Assume that you want to keep a record of average temperature of months, for the years 2003 through 2007 inclusive. Which of the following definitions best captures this data?
   (a) `float atemp[5][12];`
   (b) `float atemp[2007][12];`
   (c) `float atemp[5,12];`
   (d) `float atemp[4][12];`

3. (5 points) Which of the following is an INVALID declaration of an array as a formal parameter of a function?
   (a) `float a[4,5];`
   (b) `int a[];`
   (c) `int a[][12];`
   (d) `char s[3][5][7];`

4. (5 points) Which of the following best describes what the function below does?

```
float f (float a[], int n)
{  int i;
   float r = a[n-1];
   for (i=2; i<=n; i++)
     if (a[n-i]) > r) r=a[n-i];
   return r;
}
```

   (a) Finds the minimum value in a float array of size `n`.
   (b) Finds the number of elements in a float array `a`.
   (c) Finds the maximum value in a float array of size `n`.
   (d) Finds the index of the largest element in a float array.

5. (5 points) The following declaration produces
   `float a2[3][4][5];`
   (a) an array of 60 float numbers.
   (b) a compiler error.
   (c) an array of 2 rows of floats.
   (d) an array containing the integers 3, 4 and 5.

6. (5 points) The following declaration produces
   `float a[]={1.23,4.56};`
   (a) a compiler error.
   (b) an array of 2 float numbers.
   (c) an array of 0 rows of floats.
   (d) an array containing 6 elements.

7. (5 points) Which of the following best describes what the function below does?

```
int g (char s[], char t[])
{ int i,j,k;
  for (i=0; s[i] != '\0'; i++) {
    for (j=i, k=0;
         t[k] != '\0' && s[j]==t[k];
         j++, k++);
    if (t[k] == '\0') return (i);
  }
  return (-1);
}
```

   (a) Always returns -1.

   (b) Returns the starting index of string `t` in string `s`. Returns -1 if `t` is not a substring of `s`.

   (c) Returns the starting index of string `s` in string `t`. Returns -1 if `s` is not a substring of `t`.

   (d) Returns the index of string `t` that contains the end of string indicator `'\0'`, -1 if there is no `'\0'`.

8. (5 points) What is the value printed by the program fragment below?

```
int k=4;
int x=6;
int f(int x)
{ printf("%d",x);
  return (0);
}
main()
{
   f(k);
}
```

   (a) 4

   (b) 5

   (c) 0

   (d) 6

9. (5 points) What is wrong with the following fragment?

```
void func (int a, int b, int c)
{ int f(int a, int b, int c)
        {return (a+b+c);}
}
```

   (a) Variables `a,b,c` are declared twice.

   (b) The functions `func` and `f` have different return types.

   (c) `func` has no `return` statement.

   (d) Function `f` is defined inside `func`.

10. (5 points) What is true of the C string `"yes"` ?

   (a) Its contents is the same as array `['y','e','s']`.

   (b) It is equivalent to the logical value of `1`.

   (c) Its contents is the same as array `['y','e','s','\0']`.

   (d) None of the above.

11. (5 points) What can be said about the C loop `for(expr1; expr2; expr3) statement;` ?

   (a) `expr1` cannot be of type `float`.

   (b) It is equivalent to
   `expr1; while (expr2) {statement; expr3;}`

   (c) `expr3` cannot be empty.

   (d) `expr2` cannot be empty.

12. (5 points) Which of the following is true of C's program structure?

   (a) There can't be global variables.

   (b) There can be more than one `main` function.

   (c) There can be more than one body for some functions.

   (d) There can be functions with no arguments.

QUESTIONS 13–17 ARE ON NEXT PAGE.

13. (8 points) Given the declarations

```
int a[] = {10,15,4,25,3,-4};
int *p = a;
```

What is the result of each of the following expressions? Consider each case on its own, not sequentially. Write down the answers in the box on the right (2 points each).

(a) `*p`

(b) `a[5]`

(c) `++*p`

(d) `*(p++)`

14. (8 points) A leap year is a year which is divisible by 4 but not by 100, except that years divisible by 400 *are* leap years. For example, 2008 is a leap year (February is 29 days this year), year 1200 was leap too, but not the years 1000 or 2007.

Assume that the following fragment is supposed to print `yes` if the variable `year` holds a leap year, `no` otherwise. Write down the missing SINGLE expression below (in dots) to do that.

```
main()
int year;
{
  scanf("%d",&year);
  if (...) printf("yes");
  else printf("no");
}
```

ANSWER:

15. (8 points) Consider the following C function. Its purpose is to see if string `s1` is shorter than the string `s2`. It is supposed to return 1 if true, 0 otherwise. ONE line of code is missing from the function, indicated by dots. Write down that line in the box provided.

```
int shorter (char *s1, char *s2)
{ char *p1 = s1, *p2= s2;
  while (*p1) p1++;
  ...
  return (p1-s1 < p2-s2);
}
```

ANSWER:

16. (8 points) Imagine that we keep the average rainfall in Turkey for the years 2000-2004 in the array `rfall`. The fragment below is intended to output all years in the database that received more rainfall than an amount input by the user, in the variable `user`. Write down the missing if statement (shown in dots) to do this task. Note that years are to be printed for example as 2003, 2001 etc., not as 3 or 1.

```
#define BASEYEAR 2000
#define FYEAR 2004
float rfall[] = {233.5, 367.5, 270.35,
                 467.4, 314.2087}
main()
{ float user; int i;
  scanf("%f",&user);
  for (i=BASEYEAR; i<=FYEAR;i++)
    ...
}
```

ANSWER:

17. (8 points) A function is called *recursive* if it can call itself. A recursive function must have stopping condition(s) to avoid infinitely many calls to itself. For example, the following function is meant to compute the factorial $n! = n \times (n-1) \times \cdots 1$ recursively. The base case of the formula, which is $0! = 1$ is the stopping condition. $n!$ is mathematically recursive too, since $n! = n \times (n-1)!$ for $n \geq 1$.

```
int nfac(int n)
{
  if (n==0) return (1);
  else return (n*nfac(n-1));
}
```

However, the `nfac` function above assumes that `n` is always greater than or equal to zero. Since factorial is not defined for negative integers, we should return an error value, say -1, for negative numbers. Rewrite the function above, again recursively, with minimal addition of statements to avoid recursion if `n` is negative.

ANSWER:

THE REMAINDER OF THIS PAGE IS

INTENTIONALLY LEFT BLANK