ESC101: Fundamentals of Computing(Major Quiz 1)

02 September, 2014

Question	Points	Score
1	10	
2	12	
Total:	22	

Instructions

1. Read these instructions carefully.
2. Write you name, section and roll number on all the pages of the answer book.
3. Write the answers cleanly in the space provided. There is space left on the back of the answer book for rough work.
4. Do not exchange question books or change the seat after obtaining question paper.
5. Using pens (blue/black ink) and not pencils. Do not use red pens for answering.
6. Even if no answers are written, the answer book has to be returned back with name and roll number written.

Helpful hints

1. The questions are *not* arranged according to the increasing order of difficulty. Do a quick first round where you answer the easy ones and leave the difficult ones of the subsequent rounds.
2. For fill in the blanks type of questions, read the comments in the code. They usually have helpful remarks.

Name: Section: Rollno:

Question 1. (10 points) My friend told me that **every positive even number greater than 4** is the sum of two **ODD prime numbers**. Example: $28 = 5 + 23$. I want to verify this conjecture by writing a C program.

Complete the following program to print the two ODD prime numbers that sum up to a given even integer ≥ 4. Assume the existence of following functions:

```
int isOdd(int N);  /* returns 1  if N is odd,  0 otherwise */
int isPrime(int N); /* return 1 if N is prime, 0 otherwise */
```

Note that there are **7 blanks** that you need to fill. (Hint: Read the comments to get some help.)

```
#include <stdio.h>

int isOdd(int);
int isPrime(int);

int main()
{
    int num, prime1, prime2;
    scanf("%d", &num);
    /*Check num is even and greater than 4. If not return with error*/
    if ( ____________________________________) {
        printf("ERROR: expected even number greater than 4.\n");
        return -1;
    }

    /*Generate goldbach pair. Note that prime components must
     * be "odd", so we can take advantage of it while incrementing */
    prime1 = 3;
    while (prime1 <= _____________) {
          prime2 = ____________________________;
          if (_____________________________________)  {   break; }
          prime1 = _____________;
    }

    /* The conjecture can FAIL !! */
    if (_____________________________________) {
          printf("Conjecture failed\n");
    } else {
          /* print the two factors in nice form, for e.g.
             28 = 5 + 23 */
          printf("%d = %d + %d\n", num, ____________________________);
   }
   return 0;
}
```

Solution:

```
#include <stdio.h>
int isOdd(int);
int isPrime(int);

int main()
{
    int num, prime1, prime2;
    scanf("%d", &num);
    /*Check num is even and greater than 4. If not return with error
       */
    if (num <= 4 || isOdd(num)) {  //###[ Marks: 1 + 1 ]
        printf("ERROR: expected even number greater than 4.\n");
        return -1;
    }

    /*Generate goldbach pair. Note that prime components must
     * be "odd", so we can take advantage of it while incrementing
       */
    prime1 = 3;
    while (prime1 <= num/2) {  //###[ Marks: 1 ] 0.5 for num
          prime2 = num - prime1;  //###[ Marks: 1]
          if (isPrime(prime1) && isPrime(prime2)) {
               //###[ Marks: 2 ] no partial marks
                    break;
          }
          prime1 = prime1 + 2;  //###[ Marks: 1] 0.5 for prime1 + 1
    }

    /* The conjecture can FAIL !! */
    if (prime1 > num/2) { //###[Marks: 1], 1 for num if used in for
       loop as well
          printf("Conjecture failed\n");
    } else {
          /* print the two factors in nice form, for e.g.
             28 = 5 + 23 */
        printf("%d = %d + %d\n", num, prime1, prime2);
        //###[Marks: 2], order not important
   }
     return 0;
}
/* -------------- NOT PART OF EXAM --------------------------*/
int isOdd(int n) {return (n%2 != 0);}

int isPrime(int n)
{
    int i;
    if (!isOdd(n)) return n==2;
    for (i = 3; i*i < n; i+=2) {if (n%i == 0) return 0;}
```

```
46      return 1;
47 }
```

Question 2. (12 points) The program given next is a partially filled program that computes the *median* of three uppercase characters `c1`, `c2` and `c3` given as input, and prints **VOWEL** if it is a vowel, otherwise it prints **CONSONANT**.

For example, if `c1='A'`, `c2='E'` and `c3='W'` , then median is `'E'`, so the output is **VOWEL**.

Fill in the missing blanks and complete the program. Note that there are a total of **12 blanks** that you need to fill. Assume that the input characters will always be in the range `'A'...'Z'`.

(Hint: Read the comments to get some help.)

Name: Section: Rollno:

```
#include <stdio.h>
char min(char a, char b){    //Computes minimum of its arguments
    if(_____________) return a;
    else return b;
}

char max(char a, char b){    //Computes maximum of its arguments
    if(_____________) return b;
    else return a;
}

int main(){
    char c1, c2, c3, median;

    scanf("%c", &c1);
    scanf("%c", &c2);
    scanf("%c", &c3);

    if(max(c1,c3) == c1){
        if(max(c3,c2) == _____________)
            median = c3;
        else
            median = min(_____________,_____________);
    }
    else{
        if(min(c3,c2) == c3)
            median = _____________;
        else
            median = max(_____________,_____________);
    }

    switch (_____________) {
        default:   _____________________________;     _____________;
        case 'A':
        case 'E':
        case 'I':
        case 'O':
        case 'U': _____________________________;
    }
    return 0;
}
```

Solution:

```
#include <stdio.h>
char min(char a, char b){    //Computes minimum of its arguments
    if(a<b) return a;  //###[Marks: 1]
    else return b;
}

char max(char a, char b){    //Computes maximum of its arguments
    if(a<b) return b;  //###[Marks: 1]
    else return a;
}

int main(){
    char c1, c2, c3, median;

    scanf("%c", &c1);
    scanf("%c", &c2);
    scanf("%c", &c3);

    if(max(c1,c3) == c1){
        if(max(c3,c2) == c3)  //###[Marks: 1]
            median = c3;
        else
            median = min(c1, c2); //###[Marks: 1+1]
    }
    else{
        if(min(c3,c2) == c3)
            median = c3; //###[Marks: 1]
        else
            median = max(c1, c2); //###[Marks: 1+1]
    }

    switch (median) {
        default:   printf("CONSONANT"); break;  //###[Marks: 1 + 1]
        case 'A':
        case 'E':
        case 'I':
        case 'O':
        case 'U': printf("VOWEL");  //###[Marks: 1]
    }
    return 0;
}
```