

Name:

Section:

Rollno:

ESC101: Fundamentals of Computing (Mid Semester Exam A)

03 March 2017 (8:00 - 10:00 AM)

Total Number of Pages: 10

Total Points 200

Instructions

1. Write you name, section and roll number on all the pages of the answer book, **including the ROUGH pages**.
2. Write the answers cleanly in the space provided. Space is given for rough work in the answer book.
3. Even if no answers are written, the answer book has to be returned back with name and roll number written.
4. Sign the attendance sheet.
5. Assume `#include<stdio.h>` in the programs.
6. Fill in the blanks: Do **NOT** use ternary operators, single line while loops etc. Only simple expressions are allowed. Every blank has 5 points.
7. Read comments given inside codes for help.

Question	Points	Score
1	50	
2	50	
3	100	
Total:	200	

I PLEDGE MY HONOUR THAT DURING THE EXAMINATION I HAVE
NEITHER GIVEN NOR RECEIVED ASSISTANCE.

.....
Signature

Question 1.

- (a) (5 points) What is the output of this program?

```
1 int main() {  
2 char a[] = {'G','a','l','a','x','y','\0'};  
3 a[0] = (int)((char)((int)('G' - 'A' + 'a')));  
4 printf("%s",a); }
```

Solution: galaxy

- (b) (5 points) Calculate the number of times the statement (
- $\text{sum} = \text{sum} + i$
-) is executed in this program. What is the polynomial that you obtained (in terms of
- N
-)?

```
1 for(i=0; i<N; i++)  
2     for(j=i; j<N; j++)  
3         sum = sum + i;
```

Solution: $N(N+1)/2$

- (c)
- Circle**
- True or False

- (a) (3 points) A function must always have a return value. (True / False)
- (b) (4 points) Using scanf with %s ensures that an EOF is added to the array after the string has been read. (True / False)
- (c) (4 points) It is possible to swap the value of two integer variables without using an extra variable. (True / False)

Solution: False, False, True

- (d) (5 points) What is the output of this program?

```
1 int x = 10;  
2  
3 int foo(int a){  
4     static int x = 1;  
5     x = x+1;  
6     return a+x;  
7 }  
8  
9 int bar(int a){  
10    x = x+1;  
11    return a+x;
```

Name:

Section:

Rollno:

```
12 }
13
14 int main(){
15     printf("%d",foo(foo(5)));
16     printf("%d",bar(5));
17     return 0;
18 }
```

Solution: 10,16

(e) (12 points) What is the output of the following program?

```
1 int x = 0;
2
3 void foo(int a[], int b){
4     x = x+1;
5     b = x+10;
6     a[x] = b;
7 }
8
9 int main(){
10     int a[] = {5,4,3,2};
11     int i=0,b=0;
12     foo(a,b); foo(a,b);
13     for (i=0;i<=4;i++)
14         printf("%d",a[i]);
15
16     printf("%d",b);
17     return 0;
18 }
```

Solution: 5,11,12,2<Garbage Value>,0 (2 marks each) Mentioning the possibility of a runtime error is okay instead of Garbage Value.

Name:

Section:

Rollno:

(f) (12 points) **Switch Case:** Please write the outputs for the following cases.**Marking Scheme:** +3 for each correct output, No negative marks.

```

1 int main() {
2     int i, j;
3     scanf("%d", &j);
4     switch (i = ((i = j) + 1)) {
5         case 0:
6             printf("LINE 1\n");
7             break;
8         case 1:
9             printf("LINE 2\n");
10        case 2:
11            printf("LINE 3\n");
12            break;
13        case 3:
14            printf("LINE 4\n");
15        default:
16            printf("LINE 5: i=%d\n", i);
17    }
18    return 0;
19 }

```

Input	Predicted Output
2	
0	
-1	
1	

Solution:

Input	Predicted Output
2	LINE 4 LINE 5: i=3
0	LINE 2 LINE 3
-1	LINE 1
1	LINE 3

Name:

Section:

Rollno:

Question 2. (a) (20 points) Following code finds the greatest integer less than $\log_m y$ given $m, y, m \neq 1$ (the floor of $\log_m y$). For example, $y=30$ and $m=2$ should output 4. Please fill in the blanks to complete the program.

```
1 int log_counter(int y, int m){
2     _____ counter = 0;
3     if (_____ != 0)
4         counter = counter + 1;
5
6     return counter;
7 }
8
9 int main(){
10     int y, m;
11     scanf("%d %d", &y,&m);
12     int old_counter = _____;
13     int counter = log_counter(y,m);
14
15     while(counter != old_counter){
16         old_counter = counter;
17         y = _____;
18         counter = log_counter(y,m);
19     }
20
21     printf("%d",counter);
22     return 0;
23 }
```

Solution:

```
1 (5) static int counter = 0;
2 (5) if (y/m != 0)
3 (5) int old_counter = ANYTHING SYNTACTICALLY VALID; (except 1)
4 (5) y = y/m;
```

ROUGH WORK

Name:

Section:

Rollno:

- (b) (30 points) **Conjecture:** 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:

```
1 int notEven(int N); /* returns 0 if N is even, 1 otherwise */
2 int isPrime(int N); /* return 1 if N is prime, 0 otherwise */
```

Read the comments to get some help.

```
1 /*ASSUME that these functions have been implemented already*/
2 /*These incomplete definitions will NOT lead to errors*/
3 int notEven(int);
4 int isPrime(int);
5
6 int main()
7 {
8     int num, prime1, prime2;
9     scanf("%d", &num);
10    /*Check num is even and greater than 4. If not return with
        error*/
11    if ( _____ ) {
12        printf("ERROR: expected even number should be greater
            than 4.\n");
13        return -1;
14    }
15
16    /*Generate pair. Note that prime components must
        * be "odd", so we can take advantage of it while
        incrementing */
17    for ( prime1 = 3; prime1 <= _____; prime1 =
        _____ ) {
18        prime2 = _____;
19        if ( _____ ) {
20            break;
21        }
22    }
23
24    /* The conjecture can FAIL !! */
25    if ( _____ ) {
26        printf("Conjecture failed\n");
27    } else {
28        /* print the two factors in nice form, for e.g.
            28 = 5 + 23 */
29        printf("%d = %d + %d\n", num, prime1, prime2);
30    }
31    return 0;
32 }
33
34 }
```

Solution:

```
1 #include <stdio.h>
2 int notEven(int);
3 int isPrime(int);
```

```
4
5 int main()
6 {
7     int num, prime1, prime2;
8     scanf("%d", &num);
9     /*Check num is even and greater than 4. If not return with
       error*/
10    if (num <= 4 || notEven(num)) { //###[ Marks: 2 + 3 ]
11        printf("ERROR: expected even number should be greater
               than 4.\n");
12        return -1;
13    }
14
15    /*Generate pair. Note that prime components must
       be "odd", so we can take advantage of it while
       incrementing */
16    for ( prime1 = 3; prime1 <= num/2 ; prime1 = prime1 + 2 )
17    {
18        //###[ Marks: 5 + 5 ] ; 2 for num; 2 for prime1 + 1
19        prime2 = num - prime1; //###[ Marks: 1]
20        if (isPrime(prime1) && isPrime(prime2)) {
21            //###[Marks: 5] no partial marks
22            break;
23        }
24    }
25
26    /* The conjecture can FAIL !! */
27    if (prime1 > num/2) { //###[Marks: 5], 5 for num if used
28        printf("Conjecture failed\n");
29    } else {
30        /* print the two factors in nice form, for e.g.
31           28 = 5 + 23 */
32        printf("%d = %d + %d\n", num, prime1, prime2);
33        //###[Marks: 5], order not important
34    }
35    return 0;
36 }
37 /* ----- NOT PART OF EXAM -----
   */
38 int notEven(int n) {return (n%2 == 1);}
39
40 int isPrime(int n)
41 {
42     int i;
43     if (!notEven(n)) return n==2;
44     for (i = 3; i*i <= n; i+=2) {
45         if (n%i == 0)
46             return 0;
47     }
48     return 1;
49 }
```

Question 3. Little Shivam is very upset that he can't add, multiply very big integers using types int or even long long int. Rohan being the sweet guy he is, helps Shivam understand how to add/multiply two very large numbers (that has around 200 digits). He has provided Shivam with an incomplete code sketch of the functions add(), mult() and multiply() that accomplish tasks as specified in the comments. Kindly, complete the code sketch.

(a) (60 points) This code will help Shivam add two very big numbers.

```

1  /* function add(), adds two very large numbers.
2   s1 and s2 are strings that store very large numbers. At the end
   of function call, s1 should store the sum of s1 and s2.
3
4   Say, s1 = "1000000000000000000000", s2 = "4382748737483".
5   After function call, s1 = "1000000004382748737483" */
6
7  void add(char s1[], char s2[]){
8
9      int n1[1008], n2[1008], sum[1008], i, j, k;
10
11     // Convert s1 into integer digits and store them in n1[]
12
13     for(i=0; s1[i]!='\0'; i++){
14         n1[i] = _____ }
15
16     // Do something similar for the string s2 using n2[]
17
18     for(j=0; s2[j]!='\0'; j++){
19         _____ }
20
21     int carry = 0;
22
23     // Add n1[] and n2[] using sum[] and carry (for carry-over)
24
25     for(_____; i>=0 && j>=0; _____){
26         sum[k] = _____
27         carry = _____
28         k++; }
29
30     // Strings s1 and s2 might not be of equal size. Identify those
       conditions, also update the variable 'carry'.
31
32     while(_____){
33         sum[k] = (n1[i] + carry)%10;
34         _____
35         k++;
36         i--; }
37
38     while(_____){
39         sum[k] = (n2[j] + carry)%10;
40         _____
41         k++;
42         j--; }
43
44     if(_____)
45         sum[k++] = carry;
46
47     // sum[] stores the sum in reverse order. Store reverse in s1[].
```


Name:

Section:

Rollno:

```

48
49         for(-----,i=0; k>=0; k--,i++){
50             s1[i] = (char) (sum[k] + '0');           }
51
52         s1[i] = '\0';
53     }

```

Solution:

```

1 (5) s1[i] = '0';
2 (5) n2[j] = s2[j] - '0';
3 (5) k=0,i--,j--
4 (5) i--,j--
5 (5) (n1[i] + n2[j] + carry)%10;
6 (5) (n1[i] + n2[j] + carry)/10;
7 (5) i>=0
8 (5) carry = (n1[i] + carry)/10;
9 (5) j>=0
10 (5) carry = (n2[j] + carry)/10;
11 (5) carry > 0
12 (5) k--

```

- (b) (40 points) Complete the following code for functions mult() and multiply(). While filling blanks for multiply(), assume that the functions add()/mult() work correctly.

```

1  /* function mult(), multiplies a very large number with a single
2     digit number.
3
4  s1 is a string that stores a very large number and s2 is a
5     character from [0-9]. At the end of function call s1 should
6     store the product of s1*s2.
7
8  Say, s1 = "1000000000000000000000", s2 = '9'.
9  After function call, s1 = "9000000000000000000000". */
10
11 void mult(char s1[], char s2){
12
13     int i,size,k;
14     for(size=0; s1[size] != '\0'; size++){
15         int num[1008],carry=0;
16         for(k = size-1,i=0; k>=0; k--,i++){
17             num[i] = -----
18             carry = ----- }
19
20     if(carry > 0)
21         -----
22
23     for(k=i-1,size=0; k>=0; k--,size++){
24         s1[size] = (char) (num[k] + '0');    }
25
26     s1[size] = '\0';
27 }
28
29 /* function multiply(), multiplies two very large numbers.
30 s1 and s2 are strings that store very large numbers. At the end
31 of function call s1 should store the product of s1 and s2.

```

Name:

Section:

Rollno:

```

27
28 Say, s1 = "9892787", s2 = "1000000000000000000000".
29 After function call, s1 = "9892787000000000000000000000".
30
31 Also, notice that the function uses library functions given in
   the header file <string.h>
32 (1) strcpy(a,b) : Copies string b into string a
33 (2) strcat(a,b) : Concatenates string a with string b
   */
34
35 void multiply(char s1[], char s2[]){
36     char ans[1008];
37     strcpy(ans,s1);
38     int size1, size2,i;
39     for(size1=0; s1[size1]!='\0'; size1++);
40     for(size2=0; s2[size2]!='\0'; size2++);
41     mult(_____);
42     char zeros[1008];
43     strcpy(zeros,"0");
44     for(_____; i>=0; i--){
45         char temp[1008];
46         strcpy(temp,s1);
47         mult(_____);
48         strcat(temp,zeros);
49         add(_____);
50         strcat(_____);
51     }
52     for(i=0; ans[i]!='\0'; i++){
53         s1[i] = ans[i];
54     }
55     s1[i] = '\0';
56 }

```

Solution:

```

1 (5) (((s1[k] - '0') * (s2 - '0')) + carry) % 10;
2 (5) (((s1[k] - '0') * (s2 - '0')) + carry) / 10;
3 (5) num[i++] = carry;
4 (5) ans,s2[size2-1]
5 (5) i = size2-2
6 (5) temp,s2[i]
7 (5) ans,temp
8 (5) zeros,"0"

```