Computer Science Foundation Exam

COMPUTER SCIENCE I

Section A

No Calculators!

GRADING SCHEME

Name: _____

SSN: _____

Score: ____/50

In this section of the exam, there are Three (3) problems

You must do all of them.

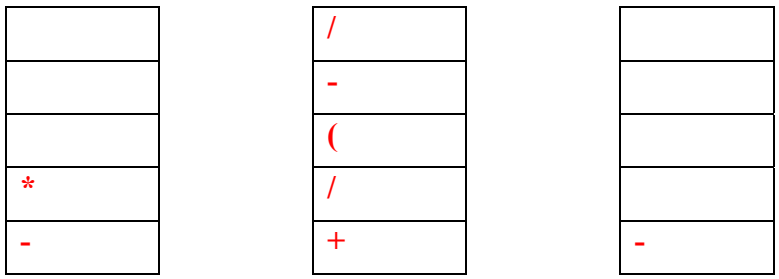
The weight of each problem in this section is indicated with the problem. The algorithms in this exam are written in C programming language notation. Partial credit cannot be given unless all work is shown.

As always, be complete, yet concise, and above all be neat. Credit cannot be given when your results are unreadable.

1. Convert the following infix expression to postfix, showing the values in the operator stack at each indicated point in the infix string (points 1, 2, and 3), in the boxes below. You may add more boxes for each position if necessary to do so. **[12 points]**

A – D – L * T [1] **+ B / (K – F * P /** [2] **Y) –** [3] **E + M**

GRADING SCHEME:
Box 1, 2, 3 **[3 points each]**
Resultant expression **[3 points]**



1 2 3

Resulting Postfix Expression : A D – L T * – B K F P * Y / – / + E – M +

2. What is the output of the following program? If you think the program is not going to run because of some errors, point out those erros. **[10 points]**

```
#include <stdio.h>
int alpha (int *px , int y ,  int x[],  int *pz);

int main()
{
      int i, a= 2, b =10 ,c = 9;
            int d[8] = {2,3,4,5};
            int *pc, *pb;
            pc = &c;
      pb = &b;
      a = alpha(pc, a ,d, pb);
      printf("\n%d, %d,  %d\n", a, b, c);
      for (i =0; i<8; i++)
      printf("%d, ", d[i]);
      printf("\n");

      return 0;
}
int alpha(int *px,  int y,  int x[],  int *pz)
{
      int i,b = 5;
     int *p,*pn, *qn;
      printf("\n %d, %d, %d",*px, y, *pz);
      printf("\n");
      for (i =0; i<8; i++)
            printf("%d, ", x[i]);
      printf("\n");
      p = px;
      pn = x+1;
      qn =&x[2];
      *pn = 15;
      *qn = 8;
      pn++;
      *pn=12;
      *(qn+3) = 20;
      pn++;
      pn[1] = 14;
      qn =pn+1;
      (*qn)++;
      *p = y + *p;
      *pz = 2 * b;
      return (*pz + *px - y );
}
```

Answer:

9, 2, 10 **[2 points]**
2, 3, 4, 5, 0, 0, 0, 0 **[2 points]**
19, 10, 11 **[2 points]**
2, 15, 12, 5, 15, 20, 0, 0 **[4 points]**

3. Verify that the following code generates the Fibonacci sequence for a particular value of n. Find its time complexity. **[8 points]**

```
int Addfib(int k, int fterm,  int sterm))
```

```
int fib(int k);
main (  )
{
     int i,n, fib_number;
     scanf("%d", n );
      for (i= 0; i<=n ; i++ ) {
           fib_number = fib( i);
      printf("\n %d ",fib_number);
     }
}

int fib(int k)
{
    return (Addfib(k, 0, 1));
}

int Addfib(int k, int fterm,  int sterm))
{
    if (k == 0)   return (fterm);
    if (k == 1)   return (sterm);
  return Addfib(k-1,  1,  fterm+sterm  ));
}
```

Verification: Try for n = 5
= Addfib(5,0,1)
= Addfib(4,1,1)
= Addfib(3,1,2)
= Addfib(2,2,3)
= Addfib(1,3,5)
As k-1 is 1, at this call, it will return sterm, i.e. 5
= 5

For any value of n, the number of calls is n at end of which the result is available. Each call involves one add operation. Thus the time complexity of the code is O(n).

Grading Scheme: 3 points for verification
5 points for time complexity

4. Study the following code segment and indicate the time complexity using Big-O notation: **[10 points]**

```
sum = 0;
for ( jj = 2; jj <= n ; jj++ ) {
    for ( kk = 5 ; kk <= n; kk++) {
          for ( m = 1 ; m<= kk ; m++ ) {
               sum = sum + m;
           }
      }
}
```

Innermost loop is gets executed kk times for each run of second loop.

$$\sum_{jj=2}^{n} \sum_{kk=5}^{n} \sum_{m=1}^{kk} 1$$

$$\sum_{jj=2}^{n} \left(\sum_{kk=1}^{n} kk - \sum_{kk=1}^{4} kk \right)$$

$= (n - 2 + 1) . \ [\ n(n+1)/2 \ - 4(5)/2 \ ]$

$= (n - 1) [n^2 + n - 20] /2$

$= [n^3 - 2 1 n + 20] /2$

So the complexity is $O(n^3)$

Grading:
Deduct 4 points if student finds the value of sum as the complexity.
Deduct 4 points if summation of middle loop is not broken in two parts and computed properly.
Deduct 2 point if first summation is done from 1 to n instead of 2 to n.

5. The following "add" function is supposed to add two positive integers stored in linked lists listA and listB, and return the result in listC. The integers are stored in reverse order, one digit per node. Thus to add the integers 7145 and 398, listA would contain the digits 5,4,1,7 and listB would contain the digits 8,9,3 in the order shown. Each node in the linked lists has the structure shown below. Complete the missing links. **[10 points]**

```
typedef struct node {
      int    data;
      struct node *next;
} list;

list* add(list* listA, list *listB) {

  list *listC, *temp;
  int firstnode = 1;
  int data1 = 0, data2 = 0, carry = 0, result;
  listC = NULL;
```

while **((listA != NULL) && (listB != NULL))** { **[3 points]**

```
    if (firstnode) {  //form the first node of list C
      listC = (list *)malloc(sizeof(list));
      temp = listC;
      firstnode = 0;
    }


    else {   //form the other nodes
          temp -> next = (list *)malloc(sizeof(list));
                                [ 1 point ]
          temp = temp -> next;
    }

    if (listA != NULL) {
         data1 = listA -> data;
         listA = listA -> next;
    }
    else
         data1 = 0;   [ 1 point ]

    if (listB != NULL) {
         data2 = listB -> data;
         listB = listB -> next;
    }
    else
         data2 = 0;

    result = (data1 + data2 + carry)%10;
    carry = (data1 + data2 + carry)/10;   [ 2 points ]

    temp->data = result;
    temp->next = NULL;

  }// end of while loop     [ 2 points ]
```

```
  if (carry != 0) {    // get a node for the last carry
    temp -> next = (list *)malloc(sizeof(list));
    temp = temp -> next;
    temp -> data = carry;
    temp -> next = NULL;
  }

  return listC;
}
```

[1 point]