Birla Institute of Technology & Science, Pilani
Second Semester 2014-2015, CS F111 Computer Programming
Mid Semester Test (Open Book)

Date: March 10, 2015 **Time: 5.00 - 6.30 PM** **Max. Marks: 75**

[*NOTE: Answer all parts of a question collectively*]

Q1. On a Unix operating system, a user named sweta has created following two sub-directories in her home directory: Subdir_1 and Subdir_2. In Subdir_1, sweta created another subdirectory named class_programs. Also, sweta saved two files (first.c and second.c) in the directory class_programs. With respect to this, answer the following parts: **[2 + 2 + 2 = 6M]**

a) Assume that sweta is in Subdir_2 directory. Give a single UNIX command to rename the file first.c to last.c

SOL: mv ./../sub_dir1/class_programs/first.c ./../sub_dir1/class_programs/last.c Marking: 0/2 Marks.

b)Assume that sweta is in her home directory. Give a single UNIX command to open the file second.c in a vi editor.

SOL: vi ./sub_dir1/class_programs/second.c Marking: 0/2 Marks

c) Assume that sweta is in directory class_programs. The C program saved in a file second.c uses sqrt() function. For a gcc compiler, give a single UNIX command for compiling and creating an output file named test.out

SOL: gcc second.c -o test.out OR gcc second.c -lm -o test.out Marking: 0/2 Marks

Q2. Answer the following: **[3 + 4 + 4 = 11M]**

a) Let A = 1 01111110 10000000000000000000000. If A is represented as an IEEE-754 32-bit floating point representation, give its decimal equivalent.

Sign bit S = 1 ? negative number
E = 0111 1110 = 126-127=-1
Fraction is 1.1 (with an implicit leading 1) = 1 + 2^-1 = 1.5
The number is -1.5 × 2^-1= -0.75
Marking: Finding Exponent correctly (i.e. -1): 1M
Finding fraction correctly (i.e. 1.5): 1M
Finding final answer (i.e. -0.75) : 1M
There are no marks for only finding sign. However, if someone interprets sign wrongly, then deduct 1M.

b) Assuming 32-bit 2's complement number representation, do the following operations.

i) $(A3FD)_{16} + (9CE)_{16} = (?)_{16}$ ii) $(A3FD)_{16} - (9CE)_{16} = (?)_{16}$

SOL: (i) FFFF9DCB Marking: If someone writes 9DCB then give 1.5M. Otherwise, for FFFF9DCB: 2M.
(ii) FFFFAA2F Marking: If someone writes AA2F then give 1.5M. Otherwise, for FFFFAA2F: 2M.

c) Convert the following unsigned numbers from given number system to desired number system.

i) $(425)_6 = (\quad)_8$ ii) $(615)_{10} = (\quad)_7$

SOL: (i) 241 Marking: 0/2M
(ii) 1536 Marking: 0/2M

Q3. Give **the output of the following programs. Assume that necessary header files are already included.**

Marking: 0M for wrong answer and 2M for correct answer. However, we will ignore the effect of /n and /t.

Q3(a)

```
int main(){
int m[] = {1,2,3,4,5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15};
int x, y ;
for(x = 0; x < 15; x++){
   for ( y = 2; y < m[x]; y++) {
      if((m[x]%y ==0)) {
            break;
      }
   }
   if (y == m[x])   {
      printf("%d ",m[x]);
   }
 }
 return (0);
}
```

SOL: 2 3 5 7 11 13 OR Prime numbers in array.

Q3(b)

```
int main(){
 int one = 1, two = 2;
 float three = 3.5;
 printf(" one = %d,two = %d,three = %f \n", one, two, three);
 {
  int one = 10;
   int two = 20;
   printf(" one = %d,two = %d,three = %f\n", one, two, three);
   {
    int three = 100.95;
    printf(" one = %d,two = %d,three =%d\n", one, two,
  (int)three);
   }
 }
  return 0;
}
```

SOL: one = 1,two = 2,three = 3.500000
one = 10,two = 20,three = 3.500000
one = 10,two = 20,three =100
Instead of 3.500000, student may write 3.5 also.

Q3(c)

```
int main (void)
{
 int a=9, b=5;
 int n,m,p=1;
 n = a - ++b;
 m = a-- + b--;
 p *= b + 5;
 printf ("n = %d m = %d p =
%d",n,m,p);
return (0);
}
```

SOL: n = 3 m = 15 p = 10

Q3(d)

```
int main()
{
   int i,j;
   i=j=2;
   while(--i&&j++)
      printf("%d %d",i,j);
   return 0;
}
```

SOL: 1 3

Q3(e)

```
int main(void)
 {
  int a=5,flag=0,b;
  b = (!flag) || ++a;
  printf ("a = %d   b =
%d",a,b);
  return (0);
 }
```

SOL: a = 5 b = 1

Q3(f)

```
int main(void)
{
 int i;
 for(i=0;i<5;i++)
    printf("%c",'a' + i);
 return (0);
}
```

SOL: abcde

[2x6 = 12M]

Q4. Observe the program shown in FIGURE 1. You can see different function calls in it. Write the *function declaration* corresponding to each function call. Do not assume any implicit type casting or default return types here. [**6M**]

```
int main (){
 int int1, int2, int3;
 char ch1, ch2, ch3;
 float flo1, flo2, flo3;
 printf ("value = %f",fun1(ch1, int1));
 fun2(flo1, int3);
 int1 = fun3 (int1, int2, int3);
 flo2 = fun4();
return (0);
}
```

FIGURE 1

Solution: float fun1 (char ch1, int int1) {}

void fun2 (float flo1, int int3) {}

int fun3 (int int1, int int2, int int3) {}

float fun4 () {}

Marking: For each function declaration: 0/1.5M . However, we will ignore the curly braces and variables. This is to say that during declaring a function before the program, there is no need to give variable names. Rest everything should be present.

Q5. Consider the following two programs given below and their corresponding purpose. While implementing them, the programmer has done some errors. Rewrite these programs in your answer sheet by removing these errors. Do only the necessary changes in program.

[**6x2 =12M**]

Q5(a) Following is the code for sorting the numbers in increasing order.

```
int main()  {
 int a[] = {111,22,31,43,5, 61, 72, 80, 9, 10};
 int temp, pass, k, size = 10, i;
 for (pass = 1; pass <= size; pass++)
 {
  for ( k = 1; k >= pass; k++)
  {
   if (a[k] < a[k-1])
   {
     temp = a[k];
    a[k-1] = temp;
     a[k] = a[k-1];
   }
  }
 }
 for(i = 0; i < size; i++)
     printf("%d ", a[i]);
 return (0);
}
```

Q5(b) Given an array of n integers (in the range -999 to +999), following is a program to find the largest sum of consecutive elements. For example, the largest sum for the following array of 7 integers is 8 which we get from adding (2, 3,-2, 0, 5).

-10	2	3	-2	0	5	-15

```
int main(){
int data[7], index , k , l , sum =0, max_sum;
//assume 7 array elements are taken as input here
for(index=0;index<7;++index)
{
 for(k=6;k>index;--k)
 {
  for(l=index;l<=k;++l)
   sum = sum + data[k];
   if(sum>max_sum)
         max_sum=sum;
 }
 }
printf("Largest sum is %d\n",max_sum);
return (0);
}
```

Q5(a) Following is the code for sorting the numbers in increasing order.

```
int main()  {
 int a[] = {111,22,31,43,5, 61, 72, 80, 9, 10};
 int temp, pass, k, size = 10, i;
 for (pass = 1; pass <= size-1; pass++)
 {
  for ( k = pass; k >0 ; k--)
  {
   if (a[k] < a[k-1])
   {
     temp = a[k];
     a[k] = a[k-1];
     a[k-1] = temp;
   }
  }
 }
 for(i = 0; i < size; i++)
     printf("%d ", a[i]);
 return (0);
}
```

Marking: This is a program for insertion sort. The ideal solution is given above.
Correcting statements for swapping: 2M
Correcting the loop: 4M
However, student may give some other answer also. The only thing to note is that student should not write altogether new program. He has to modify the code given.

Q5(b) Given an array of n integers (in the range -999 to +999), following is a program to find the largest sum of consecutive elements. For example, the largest sum for the following array of 7 integers is 8 which we get from adding (2, 3,-2, 0, 5).

-10	2	3	-2	0	5	-15

```
int main()
{
int data[7], index , k , l , sum =0, max_sum=-9999;
for(index=0;index<7;++index)
{
 for(k=6;k>index;--k)
 {
  sum=0;
  for(l=index;l<=k;++l)
   sum = sum + data[l];
  if(sum>max_sum)
         max_sum=sum;
 }
}
printf("Largest sum is %d\n",max_sum);
return (0);
}
```

Marking:
2M each for 3 changes. In the second for loop, if the student has changed the condition to k >= index, then also it is right.

Q6. Rewrite the program written in FIGURE 2 using only do-while loops; and in FIGURE 3 using only for loop. [**4+4 = 8M**]

Sol:

```
int main(void)
{
 int i, j;
 i=0;
 do
 {
  j=1;
  do
  {
   printf ("%d %d \t",i,j);
   j++;
  }while (j<40);
  ++i;
 }while (i<10);
}
```

FIGURE 2

Marking: 2M each for two loops.

Sol:

```
int main(void)
{
 int num;
 printf ("Enter a positive integer");
 scanf ("%d",&num);
 for (;num>0;num /= 10)
   printf ("%d",num%10);
 return 0;
}
```

FIGURE 3

Marking: 4M for correct answer. This program behaves differently from the given one when num<0. But, since in printf statement it is given to "Enter a positive integer" ,we are ruling out that possibility.

```
int main (void){
 int num;
 printf ("Enter a positive integer");
 scanf ("%d",&num);
 do {
  printf ("%d  ",num%10);
  num /= 10;
  } while (num > 0);
  return (0);
}
```

FIGURE 3

```
int main(void)
{
  int i, j;
  for(i=0;i<10; ++i)
   for(j=1; j<40; j++)
    printf("%d %d \t", i,j);
  return (0);
}
```

FIGURE 2

Q7. Write the following programs. Your programs should be neat, clear, and understandable.

(a) For some mechanical system, force p is expressed as a function of time t by:

$$p(t) = \begin{cases} 20 & \text{if } t = 1 \\ 4\,p(t-1) & 1 < t <= 6 \\ p(t-2) + 5 & \text{if } t > 6 \end{cases}$$

Write a program (*without using recursion*) to take time t as input and print the corresponding force. [**12M**]

Solution:

```
int main ()
{
 int t, force, temp;
 printf ("Enter time\n");
 scanf ("%d",&t);
 int arr[t+1]; //array to store intermediate values. t[0] does not carry any meaningful value.
 temp = 1;
 while (temp <= t)
 {
  if (temp == 1)
     arr[temp] = 20;
  if (temp > 1 && temp <= 6)
     arr[temp] = arr[temp-1]*4;
  if (temp > 6)
     arr[temp] = arr[temp-2] + 5;
  temp++;
 }
```

```
 printf ("Force = %d\n",arr[temp-1]);
}
```

Marking Scheme: 1M for taking time t as input.

2M for the case when t = 1

3M for the case when 1 < t <= 6

6M for the case when t > 6

For the case when t <= 6, there is no need to save the previous values of the computation. But when t > 6, we have to save the immediate previous value and previous to previous value of the computation. In the solution given, the already computed values are stored in array. Student may only store the previous two values of computation. So, 6M for all this logic when t > 6.

(b) You are given a list of **n** positive integers in "almost" sorted order (increasing) where every number is in sorted order except one. For example let n = 5 and the list of elements is 2 4 6 3 7. Here only 3 is the element which is out of order. Given such a list, write a program to find the element which is out of order. Your program should:

a) Reading the input elements of the array from the user.
b) Find the element which is out of order.
c) Print the element which is out of order and its position in the array. [**8M**]

Solution:

```
#include <stdio.h>

int main()
{
   int size, key = -1, i;
   scanf("%d", &size);
   int A[size];
   for(i = 0; i < size; i++)
   {
       scanf("%d", &A[i]);
   }

   if(A[0] > A[1])
         key = 0;
   else if(A[size-1] < A[size-2])
            key = size-1;
   else{
      for(i = 1; i <= size-2; i++)
      {
          if((A[i] > A[i+1])|| (A[i] < A[i-1]))
                key = i;
      }
   }

   printf("The key is %d at position %d\n", A[key], key);
     return 0;}
```

Marking:

+1 for taking size of array and declaring the array correctly.

+1 for reading in the values from the user correctly.

+1 for checking the first element is not out of order.

+1 for checking the last element is not out of order.

+3 for implementing the logic to find the remaining elements are no out of order.

+1 for printing the array elements correctly.