Université d'Ottawa
Faculté de génie

École d'ingénierie et de
technologie de l'information

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Information
Technology and Engineering

GNG1101/1501 **Examen Final Examination** **2005**

Temps alloué: 3 heures.
Examen à livre fermé.
Les calculatrices ne sont pas permises.

Essayez de répondre à toutes les questions.
Les points alloués à chaque question sont indiqués.
Le nombre total de points pour l'examen est 100.

Répondez aux questions dans l'espace fourni.
Utilisez les deux côtés de la feuille si nécessaire.

Time allowed: 3 hours.
Closed book examination.
NO Calculator allowed.

Attempt all questions.
Questions carry the weights indicated.
The total number of points for the examination is 100.

Answer the questions in the spaces provided.
Use both sides of these sheets if necessary.

Name/
Nom: _____

Student number/
Numéro d'étudiant(e): _____

Question 1	**/30**
Question 2	**/20**
Question 3	**/25**
Question 4	**/25**
Total	**/100**

Question 1a) ***(10 points)***

Donnez la sortie générée par le programme suivant:
Give the output generated by the following program:

```
#include <stdio.h>
void Fonction(int n);

void main()
{
int n;
Fonction(5);
}

void Fonction(int n)
{
int i;
if (n==1){
          for(i=1; i<=n; i++)
          printf("*");
        }
else    {
          for(i=1; i<=n; i++)
          printf("*");
        printf("\n");
        Fonction(n-1);
        }
}
```

Answer to Question 1a), Réponse à la question 1a):

```
*****
****
***
**
*
```

2 marks per line.

Question 1b) ***(10 points)***
Soit le programme suivant:
The following program is given:

```
#include <stdio.h>

void main()
{
int val1, val2, val3;

scanf("%d",&val1);
val2=val1;
val3=val1;

    while(val1!=-1)
    {
        if (val1 > val2)
        val2=val1;
        else if (val1 < val3)
             val3=val1;

    scanf("%d",&val1);
    }

printf("%d %d",val2, val3);
}
```

Dites ce qu'affiche le programme dans chacun des cas de valeurs lues suivants:
Give the screen output for each of the following cases of values read in:

i) 1 2 3 -1 4

3 1

ii) 4 3 2 -1 0

4 2

2 marks per case.

iii) 56 20 15 70 15 -1

70 15

iv) -1

-1 -1

v) 56 20 15 -1

56 15

Question 1c) ***(10 points)***

Supposez le code suivantes: / Assume the following code:

2 marks each.

```
int *zPtr;
int *sPtr=NULL;
int integer, i;
int z[5] = {1, 2, 3, 4, 5};
sPtr = z;
```

Trouvez et expliquez l'erreur dans chacun des segments de programme qui suit.
Find and explain the error in each of the following program segments.

a) `*zPtr=1;`

zPtr has not been initialized.

b) `integer = sPtr;`

Attempts to store an address into a non-pointer variable.

c) `sPtr = z[1];`

Attempts to store an int into a pointer variable.

d)
```
for (i = -1; i <= 5; i++)
     printf("%d ", *(sPtr+i));
```

For loop goes beyond the bounds of array z.

e) `++z;`

z is the name of an array so this pointer cannot be modified.

Question 2a) (*10 points*)

Un compte bancaire est alimenté de la manière suivante:
Initialement, le compte est vide.
Au premier janvier de chaque année le client titulaire du compte dépose une certaine somme. Cette somme est la même chaque année.
Au 31 décembre de chaque année, le client perçoit un intérêt sur le solde du compte à cette date. Le taux d'intérêt est le même chaque année.

Écrivez une fonction **`calcule`** qui reçoit comme paramètres la somme déposée chaque année, le taux d'intérêt et un nombre d'années `t`. La fonction retourne le solde du compte en date du 31 décembre après `t` années.

Exemple :
Somme déposée chaque année: $100
Taux d'intérêt: 0.1

À la fin de la première année, en date du 31 décembre, le client aura:
$100 + (10% * $100)

À la fin de la deuxième année, en date du 31 décembre, le client aura:
($100 + (10% * $100) + $100) + (10% * ($100 + (10% * $100) + $100))

-

A bank account receives deposits as follows:
Initially, the account is empty.
On the first of January of every year, the account holder deposits an amount of money. The amount deposited is the same every year. On the 31st of December of every year, the client receives interest on the account balance at that date. The interest rate is the same every year.

Write a function **`compute`** that receives as parameters the amount deposited each year, the interest rate and a number of years `t`. The function returns the account balance on the 31st of December after the `t` years.

Example:
Amount deposited every year: $100
Interest rate: 0.1

At the end of the first year, on the 31st of December, the client will have:
$100 + (10% * $100)

At the end of year 2, on the 31st of December, the client will have:
($100 + (10% * $100) + $100) + (10% * ($100 + (10% * $100) + $100))

Réponse à la question 2a) / Answer to question 2a):

Prototype de la fonction **`calcule`** / Function prototype for **`compute`** (*2 points*):

```
float compute(float rate, float dep, int yr);
```

or

```
float compute(float, float, int);
```

Fonction **`calcule`** / Function **`compute`** **(***8 points***)**:

```
float compute(float rate, float dep, int yr)

{
int i;
float resul=0;

    for(i=1; i<=yr; i++)
    {
    resul=resul+dep;
    resul=resul+rate*resul;
    }

return(resul);
}
```

Question 2b) (*10 points*)

Écrivez la fonction **maxtriangle** qui trouve le maximum des éléments du triangle supérieur d'un tableau bidimensionnel N×N, c'est à dire qui trouve le maximum des éléments sur la diagonale ou au dessus de la diagonale du tableau. La fonction **maxtriangle** retourne la valeur maximale trouvée ainsi que sa position dans le tableau, c'est-à-dire, les indices de ligne et de colonne correspondants.

Exemple: Soit le tableau A suivant :

$$\begin{pmatrix} 12 & 10 & 13 \\ 1 & 23 & 14 \\ 11 & 17 & 16 \end{pmatrix}$$

La fonction **maxtriangle** trouve le maximum (et sa position) parmi les valeurs 12, 23, 16, 10, 14, 13. Le résultat est : 23 pour maximum, 1 pour indice de ligne et 1 pour indice de colonne. Si le maximum apparaît plusieurs fois, la première position du maximum est retournée.

Write the function **maxtriangle** which finds the maximum of all the elements in the upper triangle of a two-dimensional N×N array, that is, which finds the maximum of the elements on or above the diagonal of the array. Function **maxtriangle** returns the maximum value found as well as its location in the array, i.e.: its corresponding row and column indices.

Example: Consider the array A:

$$\begin{pmatrix} 12 & 10 & 13 \\ 1 & 23 & 14 \\ 11 & 17 & 16 \end{pmatrix}$$

The function **maxtriangle** finds the maximum (and its position) among the values 12, 23, 16, 10, 14, 13. The result is: maximum 23, row index 1, column index 1. If the maximum appears more than once, the location of its first occurrence is given.

Réponse à la question 2b) / Answer to question 2b):

Prototype de la fonction **`maxtriangle`** / Function prototype for **`maxtriangle`** (*2 points*):

```
void maxtriangle(int tab[][N], int *max, int *rmax, int *cmax);
```

or

```
void maxtriangle(int [][N], int *, int *, int *);
```

Fonction **`maxtriangle`** / Function **`maxtriangle`** **(***8 points***)**:

```
void maxtriangle(int tab[][N], int *max, int *rmax, int *cmax)
{
int i,j;

*max=tab[0][0];
*rmax=0;
*cmax=0;

    for(i=0; i<N; i++)
        for (j=i; j<N; j++)
           if (tab[i][j] > *max)
           {
           *max=tab[i][j];
           *rmax=i;
           *cmax=j;
           }
}
```

A possible and acceptable variant of this function returns one of the arguments using a return statement (e.g.: `return max;`) while returning the rest of the arguments as a simulated call by reference.

Question 3 ***(25 points)***

a) ***(17 points)***

Pour le code suivant, vous pouvez supposer que `a`, `b`, `aPtr` et `bPtr` sont en mémoire aux adresses 1000, 2000, 3000 et 4000, respectivement, et que `sizeof(int)` donne 4. Donnez la sortie dans les boites qui se trouvent plus bas, générée par chacun des programmes suivants.

For the following code, you may assume that `a`, `b`, `aPtr` and `bPtr` are located at addresses 1000, 2000, 3000 and 4000, respectively, and that `sizeof(int)` yields 4. Give in the boxes shown below, the output generated by each of the following programs.

```
#include <stdio.h>
void fun(int, int *);

void main()
{
 int a=2, *aPtr=&a,
     b=4, *bPtr=&b;

 printf("%u\n",&b);
 printf("%u\n",&aPtr);

 fun(a, bPtr);

 printf("%d\n", b);
 printf("%d\n", a);
}

void fun(int z, int *Ptr)
{
 *Ptr=z * ++(*Ptr);
 printf("%d\n", Ptr);
 printf("%d\n", z);
}
```

```
#include <stdio.h>
void func(int *);

void main()
{
 int a, *aPtr=&a;

 a=132;
 printf("%u\n", aPtr);
 printf("%d\n", a);
 func(aPtr);
 printf("%d\n", a);
 }
void func(int *y)
{
*y=*y+2;
y=y+2;

printf("%d\n",sizeof(int));
printf("%u\n", y);
}
```

```
#include <stdio.h>
#define M 2
#define N 3
void fcn(int, int[][N], int*);

void main()
{
 int b[M][N]={1}, i, j;

 fcn(b[1][1], b, &b[1][2]);

   for(i=0; i<=M-1; i++)
   {
     for(j=0; j<N; j++)
     printf("%d ", b[i][j]);
   printf("\n");
   }
}

void fcn(int x, int y[][N], int *z)
{
 int k;

  for(k=5; k>=0; k--)
  *(z-k)=77;

  for(k=0; k<=N-1; k++)
  y[x][k]=x;


 y[0][2]=66;
 x=88;
}
```

Réponse à la Question 3a) / Answer to Question 3a):

1 mark each.

2000 **3000** **2000** **2** **10** **2**	**1000** **132** **4** **1008** **134**	**0 0 66** **77 77 77**

3(b) ***(8 points)***

Donnez la sortie **précise** générée par le programme suivant. Utilisez le quadrillé ci-bas; chaque carré représente la position d'un caractère à l'écran.

Write down the **exact** output of the following code. Use the grid below, each square of the grid represents one character position on the screen.

```
#include <stdio.h>
#define M 4
void main()
{
   int i, j, a[M][M], temp;

   for(i=0; i<M; i++)
      for(j=0; j<M; j++)
         a[i][j]=i+j;

   for(i=0; i<M; i++)
   {
      for(j=0; j<M; j++)
         printf("%d ",a[i][j]);
      printf("\n");
   }
   printf("\n");

   for(i=0; i<M; i++)
      for(j=i; j<M; j++)
      {
         temp=a[i][j];
         a[i][j]=a[j][i];
         a[j][i]=temp;
      }

   for(j=0; j<M; j++)
   {
      for(i=0; i<M; i++)
         printf("%d ",a[i][j]);
      printf("\n");
   }
}
```

4 marks for the first set of outputs.

4 marks for the second set of outputs.

Réponse à la question 3b) / Answer to Question 3b):

```
0 1 2 3
1 2 3 4
2 3 4 5
3 4 5 6

0 1 2 3
1 2 3 4
2 3 4 5
3 4 5 6
```

Question 4 ***(25 points)***

L'échelle de notation à l'Université d'Ottawa est décrite au tableau 1. La note finale d'un cours en pourcentage est traduite à une note alpha à laquelle est associée la valeur numérique indiquée.

%	Note Alpha	Valeur Numérique
90-100	A+	10
85-89	A	9
80-84	A-	8
75-79	B+	7
70-74	B	6
66-69	C+	5
60-65	C	4
55-59	D+	3
50-54	D	2
40-49	E	1
0-39	F	0

Tableau 1

`results.dat`

C
A
A+
A+
E

Tableau 2

Les résultats d'un cours particulier sont sauvegardés dans le fichier ASCII **`results.dat`**. La structure du fichier est indiquée au tableau 2. Le fichier contient une note alpha par ligne. Il n'y a aucune indication dans le fichier sur le nombre de notes sauvegardées.

Écrivez un programme qui lit le fichier **`results.dat`** ayant la structure présentée au tableau 2. Votre programme doit utiliser un tableau unidimensionnel **`hist[11]`**. Associez **`hist[0]`** avec la note F, **`hist[1]`** avec la note E, etc… À mesure que les données sont lues, votre programme doit compter dans le tableau **`hist`** le nombre d'occurrences de chaque note. Quand toutes les notes auront été lues, votre programme doit afficher à l'écran le nombre d'occurrences de chaque note alpha, la note alpha ayant le nombre maximal d'occurrences, la note alpha ayant le nombre minimal d'occurrences, et la moyenne pondérée de la classe en utilisant les valeurs numériques données au tableau 1. (Rappel: moyenne pondérée = somme de toutes valeurs numériques divisée par le nombre de notes alpha.)

Suivez notre approche standard de résolution de problèmes.

The marks scheme of the University of Ottawa is defined in Table 1. The final percentage in a course is translated into a letter grade and each letter grade has an associated grade point value.

%	Letter Grade	Grade Point Value
90-100	A+	10
85-89	A	9
80-84	A-	8
75-79	B+	7
70-74	B	6
66-69	C+	5
60-65	C	4
55-59	D+	3
50-54	D	2
40-49	E	1
0-39	F	0

Table 1

`results.dat`

```
C
A
A+
A+
E
```

Table 2

The results of a particular course are stored in the ASCII file **`results.dat`**. The file structure is shown in Table 2. The file contains one letter grade per line. There is no indication in the file of how many grades are in it.

Write a program that reads the file **`results.dat`** with the structure presented in Table 2. Your program must use a single dimensional array **`hist[11]`**. Associate **`hist[0]`** with the grade F, **`hist[1]`** with the grade E, etc… As the data is read, the program is to count in **`hist`** the number of occurrences of each grade. When all the grades have been read, the program is to print out to the screen the number of occurrences of each letter grade, the letter grade with the maximum number of occurrences, the letter grade with the minimum number of occurences and the grade point average for the class using the grade point values given in Table 1. (Recall: grade point average = sum of all grade point values divided by the number of letter grades.)

Follow the standard approach to problem solving.

Réponse à la Question 4 / Answer to Question 4:

Énoncé du problème / Problem statement *(2 points)*

Write a program that reads an ASCII file containing letter grades and that outputs to the screen the number of occurrences of each grade, the grades that have the minimum and maximum number of occurrences, and the GPA of the class.

Description des entrées et sorties / Description inputs and outputs *(3 points)*

Inputs: ASCII file results.dat
Outputs: Number of occurrences of each grade, maximum grade, minimum grade and class gpa.

Développement de l'algorithme / Algorithm development *(5 points)*

Open the file to be processed.
If file can be opened:

Loop until the end of the file is reached:
Read in a grade from the file.
Accumulate in an array of `int`'s (`hist`) the number of occurrences of a grade.

Loop over all grades using the array hist:
If the current grade has a number of occurrences greater than the max encountered to date, then memorize the number of occurrences and the position in the array.
If the current grade has a number of occurrences less than the min encountered to date, then memorize the number of occurrences and the position in the array.

Accumulate the points.
Accumulate the total number of grades.

Compute the class gpa.
Print out to the screen the number of occurrences of each grade.
Print out to the screen the grade has the maximum number of occurrences.
Print out to the screen the grade that has the minimum number of occurrences.
Print out to the screen the class gpa.

Programme en C / Program in C *(15 points)*

```
/*****************************************/
/* Final 2005, Question 4.               */
/* Pierre Berini.                        */
/*****************************************/
#include <stdio.h>

   void main()
{
   int ctr, maxOccPos, maxOccGrade, minOccPos, minOccGrade,
       classSize=0, hist[11]={0};
   float gpaAve=0.0;
   char input[3];
   FILE *filePtr;

  if ( (filePtr=fopen("C:\\Berini\\cours\\gng1101\\Cpp\\results.dat", "r")) == NULL)
      printf("Error opening file.\n");
  else
  {

/* Read the file and parse the grades. */
            while( (fgets(input, 3, filePtr)) != NULL)
            {
                  if(input[0] == 'A')
                        {if(input[1] == '+')
                         hist[10]++;
                         else if (input[1] == '-')
                         hist[8]++;
                         else
                         hist[9]++;}

                  else if(input[0] == 'B')
                            {if(input[1] == '+')
                             hist[7]++;
                             else
                             hist[6]++;}

                  else if(input[0] == 'C')
                            {if(input[1] == '+')
                             hist[5]++;
                             else
                             hist[4]++;}

                  else if(input[0] == 'D')
                            {if(input[1] == '+')
                             hist[3]++;
                             else
                             hist[2]++;}

                  else if(input[0] == 'E')
                            hist[1]++;

                  else if(input[0] == 'F')
                            hist[0]++;
              }

/* Find the maximum and minimum occurences of the grades */
/* and the grade point average of the class.             */
         maxOccGrade=hist[0];
         maxOccPos=0;
         minOccGrade=hist[0];
         minOccPos=0;
```

```
                for(ctr=0; ctr<=10; ctr++)
                {
                  if(hist[ctr]>maxOccGrade)
                  {
                  maxOccGrade=hist[ctr];
                  maxOccPos=ctr;
                  }

                  if(hist[ctr]<minOccGrade)
                  {
                  minOccGrade=hist[ctr];
                  minOccPos=ctr;
                  }

                gpaAve=gpaAve+ctr*hist[ctr];
                classSize=classSize+hist[ctr];
                }
        gpaAve=gpaAve/classSize;

/* Print out the number of occurences of each grade. */
       printf("Number of occurences of each grade:\n");
       printf("Number of A+: %d\n", hist[10]);
       printf("Number of A : %d\n", hist[9]);
       printf("Number of A-: %d\n", hist[8]);
       printf("Number of B+: %d\n", hist[7]);
       printf("Number of B : %d\n", hist[6]);
       printf("Number of C+: %d\n", hist[5]);
       printf("Number of C : %d\n", hist[4]);
       printf("Number of D+: %d\n", hist[3]);
       printf("Number of D : %d\n", hist[2]);
       printf("Number of E : %d\n", hist[1]);
       printf("Number of F : %d\n", hist[0]);

/* Print out the grade that occured most frequently. */
        if(maxOccPos == 10) printf("A+ occured most frequently\n");
        if(maxOccPos == 9)  printf("A occured most frequently\n");
        if(maxOccPos == 8)  printf("A- occured most frequently\n");
        if(maxOccPos == 7)  printf("B+ occured most frequently\n");
        if(maxOccPos == 6)  printf("B occured most frequently\n");
        if(maxOccPos == 5)  printf("C+ occured most frequently\n");
        if(maxOccPos == 4)  printf("C occured most frequently\n");
        if(maxOccPos == 3)  printf("D+ occured most frequently\n");
        if(maxOccPos == 2)  printf("D occured most frequently\n");
        if(maxOccPos == 1)  printf("E occured most frequently\n");
        if(maxOccPos == 0)  printf("F occured most frequently\n");

/* Print out the grade that occured lest frequently. */
        if(minOccPos == 10) printf("A+ occured least frequently\n");
        if(minOccPos == 9)  printf("A occured least frequently\n");
        if(minOccPos == 8)  printf("A- occured least frequently\n");
        if(minOccPos == 7)  printf("B+ occured least frequently\n");
        if(minOccPos == 6)  printf("B occured least frequently\n");
        if(minOccPos == 5)  printf("C+ occured least frequently\n");
        if(minOccPos == 4)  printf("C occured least frequently\n");
        if(minOccPos == 3)  printf("D+ occured least frequently\n");
        if(minOccPos == 2)  printf("D occured least frequently\n");
        if(minOccPos == 1)  printf("E occured least frequently\n");
        if(minOccPos == 0)  printf("F occured least frequently\n");

/* Print out class GPA. */
      printf("Class GPA: %f\n", gpaAve);
```

```
        }
}
```