Exercises on Structures and Strings

Q. 1

Define a structure data-type named _PLAYER which has the following members:

Name: A string of 30 characters.
DOB: A string of 10 characters.
Height: A floating point number.
Weight: A floating point number.

Structure Definition

```
typedef struct{
        char Name[31];
         char DOB[11];
         float  Height;
         float   Weight;
         }  _PLAYER;
```

Q. 2

Write a main program which reads records of N players (N to be read from the keyboard) in an array of _PLAYER and prints their names and dates of birth.

Structure Definition

```
#include <stdio.h>
#define MAX_NO 100

typedef struct {
                    char Name[31];
                    char DOB[11];
                    float Height;
                    float Weight;
                     } _PLAYER;
```

Reading and Printing

```
main()
{ _PLAYER pl
  int N,i;

  printf("Give t
  scanf("%d",&

  for(i=0;i<N;i+
  {    printf("Inp

       printf("Name? \n ");
```

```
    printf("DOB as dd/mm/yyyy? \n");
    scanf(“  %[^\n]",pl[i].DOB);

    printf("Height (cm.) and Weight (kg) ? \n");
    scanf("%f%f",&pl[i].Height,&pl[i].Weight);
   }
  printf("Data read \n");
```

```
 printf("Printing Name and Dates of Births \n");
  for(i=0;i<N;i++)
   {
     printf("Player[%d]--> %s %s \n",i,pl[i].Name,pl[i].DOB);
    }
```

Q. 3

Write a function which takes an array of _PLAYER as a parameter and returns the tallest player by deleting the player from the list and adjusting the array by moving the elements upward.

Function Implementation

```
_PLAYER tallest_player(_PLAYER p[ ], int N)
 {
   int i;
   float maxHeight=-1;
   int maxPos;
 _PLAYER tmp;

 for(i=0;i<N;i++)
 {
    if (p[i].Height > maxHeight)
    {
      maxPos=i;
      maxHeight=p[i].Height;
     }
   }
```

Computing the array index for the tallest player

```
tmp=p[maxPos];
   for(i=maxPos+1;i<N;i++)
          p[i-1]=p[i];
     return(tmp);
    }
```

Return Tallest Player

Deleting a player and readjusting array

Q. 4

Use this function to print the Names of players in the descending order of their heights.

Printing in ascending order of heights

```
main()
{  _PLAYER pl
  int N,i;
  _PLAYER tal

  printf("Give t
  scanf("%d",&

  for(
  {   p

      p
      s
```

```
    printf("DOB as dd/mm/yyyy? \n");
    scanf(" %[^\n]",pl[i].DOB);

    printf("Height (cm.) and Weight (kg) ? \n");
    scanf("%f%f",&pl[i].Height,&pl[i].Weight);
   }
  printf("Data read \n");
```

```
for(i=N;i>1;i--)
 {
    tallest=tallest_player(pl,i);
     printf("%s Height=%f \n",tallest.Name,tallest.Height);
   }
   printf("%s Height=%f \n",pl[0].Name,pl[0].Height);
}
```

Q. 5

(a) Write a function *isLC( )* which takes a character as an input and returns 1 if it is a lower case alphabet else 0.

(b) Write the corresponding function *isUC( )* for checking an upper case alphabet.

Functions for checking alphabet cases

```
int isLC(char c)
{
  if((c>='a') && (c<='z'))
    return 1;
  else
    return 0;
}
```

```
int isUC(char c)
{
  if((c>='A') && (c<='Z'))
    return 1;
  else
    return 0;
}
```

Q. 6

Write a function *toLC( )* which takes a string as an input and replaces all the Upper Case alphabets to lower cases within it. The function also returns the number of changes it made.

```
int toLC(char s[ ])
{
  int i, nChange=0;

  for(i=0; s[i]!=0; i++)
  {
    if ((s[i]>='A') && (s[i]<='Z'))
     {
      s[i]=s[i]-('A'-'a');
      nChange++;
     }
   }
    return nChange;
}
```

Q. 6-a

- Write the recursive implementation of *toLC( )* for converting Upper Cases to Lower Case.

```
int toLC(char s[ ],int pos)
{
  if(s[pos]==0)
     return 0;

  else

    if((s[pos]>='A') && (s[pos]<='Z'))
     {
      s[pos]=s[pos]-('A'-'a');
      return 1+toLC(s,pos+1);
     }

    else
      return toLC(s,pos+1);
}
```

Q. 7

- Write a function *complementString( )* which complements an input string by changing the cases of its alphabets. The function also returns the number of changes.

```
int complementString(char s[ ])
{
  int i,nChange=0;

  i=0;
  while(s[i]!=0)
  {
  if (isUC(s[i]))
   {
      s[i]=s[i]-('A'-'a');
      nChange++;
   }
   else
```

```
If (isLC(s[i]))
     {
      s[i]=s[i]+('A'-'a');
      nChange++;
     };
    i++;
  }
   return nChange;
}
```

Q. 8

- (a) Write a function *absd( )* which takes a value in *double* data-type and returns its absolute value in *double*.
- (b) Write a function *expd( )* which computes e^x for an input variable x (of *double* data type) by computing the following series summation with an accuracy of .0001.

$$e^x = 1 + x/1! + x^2/2! + x^3/3! + \ldots\ldots\ldots\ldots\ldots$$

```
double absd(double x)
{
 if(x<0) return -x;
 else return x;
}
```

Factorial is not computed directly.

$X^n/n!$

```
double expd(double x)
{
 double sum,term;
 double error_bound=.0001;
 int i;

 term=1;
 sum=1;
 i=0;

 while(absd(term)>error_bound)
 {
  i++;
  term=term*x/(double) i;
  sum=sum+term;
 }

 return(sum);
}
```

Q 8 (contd.)

- Write a program which computes exponential of values (to be read from the keyboard) in an infinite loop.

```
#include <stdio.h>

double absd (double x)
{
.
.
}

double expd (double x)
{
.
```

```
main()
{
  double y,val;

 while(1)
 {
  printf("Give y: ");
  scanf("%lf",&y);

  val=expd(y);

                       xp(y)=%lf \n",y,val);
```

```
Give y: 1
y= 1.000000 exp(y)=2.718279
Give y: -1
y= -1.000000 exp(y)=0.367882
Give y: 23
y= 23.000000 exp(y)=9744803446.248880
Give y: ^C
```

Q. 9

- Write a function which returns square root of a floating point value with an accuracy of .00001, following the Newton-Raphson method.
- Newton-Raphson method:

  Solve: $f(x)=0$

  Iteration: $x_{n+1}= x_n- f(x_n)/f'(x_n)$
- Newton-Raphson method for square root:

  $f(x)=x^2-a$

  Iteration: $x_{n+1}= x_n- (x_n^2 -a)/2x_n$

```
float newton_sqrt (float x)
{
 float error,xinit,xold,xnew;
 int iter_no=0;

 if(x<=0) return (-1);

 xinit=x;
 xold=xinit;
   do {
        xnew=xold- (xold*xold-x)/(2*xold);
        error= xnew-xold;
        if(error<0) error=-error;
        xold=xnew;
        iter_no++;
      } while(error>.00001);
    return(xold);
}
```

```
main()
{
float a,root;

printf(“Input a? \n");
scanf("%f",&a);
printf(“a= %f \n",a);

root=newton_sqrt(a);
if(root>0)
printf("%f  \n",root);
}
```

Q.10

- (a) Define a structure *_PNT* which has x-coordinate and y-coordinate (both of *float* data-type) as its members for representing a point in a 2-D space.
- (b) Write a function *dist()* which takes two points and returns the distance between them.

```
typedef struct{
        float x,y;
        } _PNT;

float dist(_PNT a, _PNT b)
{
  float d;
  d=(a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
  if(d>0)
   return((float)sqrt((double)d));
  else
   return(0);
}
```

Q. 10

- (c) Define a function *checkCircle( )* which takes center and radius of a circle and a point *p* as inputs and returns -1, 0 or 1 if *p* is outside, or, on its perimeter or inside the circle respectively. Use the function *dist( ).*
- (d) Write a *main( )* function to read center and radius of a circle and a query point *q*. The program should print whether the point is outside or inside or on the circle.

```
int checkCircle(_PNT c, float R, _PNT p)
{
   float d;

   d=dist(c,p);

   if(d>R) return (-1);
   else if (d<R) return(1);
        else return(0);
}
```

```
mai
{
  _P
  flo
  int
```

```
Read the center of circle
40 50
Read the radius of circle
10
Read a point
35 25
Outside
```

```
   printf("Read the center \n");
   scanf("%f%f",&oc.x,&oc.y);
   printf("Read the radius \n");
   scanf("%f",&R);
   printf("Read a point \n");
   scanf("%f%f",&p.x,&p.y);

   flag=checkCircle(oc,R,p);
```

```
switch(flag)
   {
     case 1: printf("Inside \n");
             break;
     case -1: printf("Outside \n");
              break;
     case 0: printf("On circle \n");
             break;
     default:
             printf("?? \n");
   }
}
```

Q. 11

- (a) Write a function *mean( )* which takes an array of N floating point numbers and computes their mean.
- (b) Write another function named *sd( )* for same input for computing standard deviation. Use *mean( ).*

```
float mean(float x[ ],int N)
{
  int i;
  float sum=0;

  for(i=0;i<N;i++)
    sum+=x[i];

 return(sum/(float)N);
}
```

```
float sd(float x[ ],int N)
{
  int i;
  float sum=0, sqrsum=0;
  float mu, var, sigma;

  for(i=0;i<N;i++)
    sqrsum+=x[i]*x[i];

   mu=mean(x,N);
   var=sqrsum/(float) N-mu*mu;

   if(var>0)
       sigma=(float)sqrt((float)var);
   else
      sigma=0;

 return(sigma);
}
```

Q. 11 (contd.)

- Write a main program to read N values in an array and compute their mean and standard deviation.

```
#include <stdio.h>
#include <math.h>

#define MAX_NO 100

float mean(float x [ ],int
{
:
}

flo                         N)
{
:
}
```

```
main()
{
 int i,N;
 float height[MAX_NO];

 printf("Input N? \n");
 scanf("%d",&N);

 printf("Give %d values \n",N);
 for(i=0;i<N;i++)
  scanf("%f",&height[i]);

 printf("Mean= %f \n",mean(height,N));
 printf("S.D.= %f \n",sd(height,N));
}
```

```
Input N?
5
Give 5 values
5 10 20 30 40
Mean= 21.000000
S.D.= 12.806249
```