Dynamic Allocation of memory

Static allocation:

Array size allocation arrayd[5]

Automatic allocation:
space for variables in a function. Memory gets freed when the function returns.

Dynamic Allocation :
Acquire memory when needed during runtime.

The type void*

```
int *ip;
```
then ip points to an integer,

```
char *cp;
```
Here cp points to a character

To allocate dynamic memory to hold an integer GRADE
```
int *GRADE;
GRADE = malloc(sizeof(int));
```

To allocate dynamic memory for an array to hold n floating point variables. The following statements would do the job:

```
double *darray;
darray = malloc(n* sizeof(double));
```

Here is an example of dynamic allocation of an array. In this example, we have to store data for a number of persons, but we do not know in advance the number. The number is read from the file when the program starts running. So the memory space to store this array of structures needs to be dynamically allocated.

The Florida Lottery System

The state of Florida has a system that automatically creates a file that contains the name of each ticket purchaser along with the combination of 6 ***distinct*** numbers picked by that person. The job is to use this file in determining everyone's winnings. In particular, here is the payout for matching a certain number of values:

Numbers Matched	Winnings
3	$10
4	$100
5	$10000
6	$1000000

The program will ask the user for an input file with the data for the ticket purchases.

Then it will ask the user for the winning combination of numbers.
(When playing the Florida lottery, you must pick 6 ***distinct*** numbers from the set {1,2,3,...,52,53}.)

The user MUST enter these numbers in ascending order.
Once these have been entered, the program should print out the names of all the winners, along with how much money they have won.

It is desired that the implementation adheres to the following guidelines:

1) Use a record to store information about each ticket.
2) Dynamically allocate an array to store all the ticket information based on the first number in the file.

Input File Format

The first line will contain **n**, the total number of tickets bought.

The next 2n lines will contain information about each ticket bought.
The information for a single ticket will be on two lines.

The first of the two lines will contain the last name of the ticket buyer, followed by the first name.

The following line will contain the six integers chosen by that buyer. Here is a sample file, *input.txt*:

```
5
Dorr Robin
1 15 19 26 33 46
Mong Drian
17 19 33 34 46 47
Pate David
1 4 9 16 25 36
Bayer Lisa
17 19 34 46 47 48
Gupta Alok
5 10 17 19 34 47
```

Implementation

Steps:

1. Create a structure to hold the name of person and the numbers chosen.
2. Ask the user to supply name of the file containing ticket data.
3. Read the number of persons "n".
4. Dynamically allocate memory for "n structures"(create temporary space).
5. Read the names and numbers from the file and store in the allocated memory.
6. Call functions to match numbers of each person with the numbers decided by the lottery organizers, and declare the winners.
7. Free the temporary space created for the structure.
8.

```
#include<stdio.h>

typedef struct
{
   char first[20];
   char last[20];
   int nos[6];
}gambler;

gambler *read_file(char file[],   int *numbuyers);
void print_winners( gambler *list,  nt numtickets );
void  match_one (gambler buyer ,  int win_nos[ ] );


int main()
{
   char filename[20];
   int numbuyers, win_nos[6];
   gambler *ticket_buyers;
   printf ("\n Enter the name of the file\n");
   scanf ("%s",filename);

  // reads file information into memory and prints the winning numbers.

   ticket_buyers = read_file(filename,  &numbuyers);

   print_winners(ticket_buyers,   numbuyers);



   // free the memory space used by this dynamic allocation.

   free (ticket_buyers);
   return 0;
}

//Preconditions: file should be name of file storing the ticket buyer
//information in proper format.
//Postconditions: Returns a Pointer to the array storing all ticket buyer
//information and total number of ticket buyers.

gambler *read_file (char file[],   int *numbuyers)
{
   FILE *fp;
   int   i, j;
   gambler  *list  ;

   fp =  fopen( file,   "r");
   fscanf (   fp,  "%d",    numbuyers);

   list=(gambler*) malloc((*numbuyers)*sizeof(gambler));

    // read in individual ticket buyer information

   for (i=0;   i<*numbuyers;    i++)
      {
         fscanf(  fp,   "%s",  list[i].last );
         fscanf(  fp,   "%s",  list[i].first);

         for( j  = 0; j<6; j++)
            fscanf( fp, "%d" ,&( list[i].nos[j] ));
      }

    fclose(fp);
    return list;
```

```
}
```

// This FUNCTION looks at the LIST and gets winning numbers from the user

```
void print_winners( gambler *list, int numtickets )
{
   int i,  k , win_nos[6];

   printf("\nEnter the winning lottery nos :\n ");
   for( i=0 ;i<6 ;i++)
   scanf("%d",&win_nos[i]);
```

//for each ticket buyer , match the numbers with the winning numbers.

```
   for  ( k = 0;    k< numtickets;  k++)
      match_one( list [k ] ,  win_nos  );
}
```

// FUNCTION TO MATCH DATA

```
void  match_one (gambler buyer , int win_nos[] )
{
   int i=0 ,    j=0,    count=0;
      while(i<6 && j<6)
        {
           if( win_nos[i] <  buyer.nos[j] )
                   i++;
           else  if (win_nos[i]  >  buyer.nos[j] )
                   j++;
           else
                 {
                     count++;
                     i++;
                     j++;
                 }
        }
```

//Depending on number matched print out the result

```
     switch(count)
      {
     case 3:
        printf( "\n%s %s   matched  3 numbers and won
        $10.\n",
        buyer.first, buyer.last);
        break;

     case 4:
        printf( "\n%s %s   matched  4 numbers and won
        $100.\n",
        buyer.first, buyer.last);
        break;
     case 5:
        printf( "\n%s %s   matched  5 numbers and won
        $10000.\n",
        buyer.first, buyer.last);
        break;

     case 6:
        printf( "\n%s %s   matched  6 numbers and won
        $1000000.\n",
        buyer.first, buyer.last);
        break;
     }
}
```

Double Pointers

In this example, we want to fill up a two dimensional array with the grades obtained by students in a number of exams. The *number of students* and the *number of exams* they appeared in are not known in advance. This information is to be read from the user's file. Since *grades* is a two dimensional array, it is declared with double pointers.

```
int main()
{
        FILE* fp;
        int i,j, students, exams, **grades;

        printf("Please enter name of input file.\n");
        scanf("%s", filename);
```

//open file

```
fp = fopen(filename, "r");
if(!fp)
{
  printf("sorry, %s not opened\n", filename);
        exit(0);
}
```

//Read from the file the number of students and the number of exams they appeared in

```
fscanf(fp, "%d", &students);
fscanf(fp, "%d", &exams);
```

//allocate space for grades

```
grades = (int**)malloc(students * sizeof(int*));
for(i=0; i<students; i++)
   grades[i] = (int*)malloc(exams * sizeof(int));
```