1. Write a **C** program named **p1.c** that produces output in the following format (program output is shown in **blue**, and user input is shown in **green**, but you don't need to produce colored output!). Use dynamically allocated memory to store the **n** input data values. Your source code should include a comment that shows the command you used to create the executable program.

```
How many? Enter n: 3
Enter 3 integer data values: 1 2 3
You entered: 1 2 3
```

```c
// p1.c
// gcc -o p1 p1.c

#include <stdio.h>
#include <stdlib.h>

int main()
{
  int  i, n;
  int *data;

  printf( "How many? Enter n: " );
  scanf( "%i", &n );

  data = (int *)calloc( n, sizeof(int) );

  printf( "Enter %i integer data values: ", n );
  for( i = 0; i < n; ++i )
    scanf( "%i", &data[i] );

  printf( "You entered: " );
  for( i = 0; i < n; ++i )
    printf( "%i ", data[i] );
  printf( "\n" );

  free( data );

  return 0;
}
```

2. In a text file (not WORD, not pdf., etc) file named **answers.txt**, write a typedef for a struct named Midn that contains the following information:

First name, Middle initial, Last name, Company, CQPR

Your struct data members should be of the appropriate type.

```c
typedef struct
{
  char  fname[32]; // First name (an assumed size)
  char  mi;        // Middle initial
  char  lname[32]; // Last name (an assumed size)
  int   co;        // Company
  float cqpr;      // Cumulative QPR
} Midn;
```

3. Add these answers to your file named **answers.txt**. Assume you have these declarations: Midn *mp1, *mp2, *mp3;

Write a statement that uses malloc to allocate memory for one Midn.
Write a statement that uses malloc to allocate memory for three Midn.
Write a statement that uses calloc to allocate memory for three Midn.
Write the appropriate statements to return the memory allocated above.

```
mp1 = (Midn*)malloc(     sizeof(Midn) );
mp2 = (Midn*)malloc( 3 * sizeof(Midn) );
mp3 = (Midn*)calloc( 3,  sizeof(Midn) );
free(mp1);
free(mp2);
free(mp3);
```

4. Assume you have a text file named "data.txt" that contains *account information* in the following format:

```
3                          (First line: the number of lines of data in the file)
4305 123.45 Door           (Each subsequent line: Account number, account balance,
account last name)
1106 98.76 Gish
6816 1980.28 Stahl         ( etc ... )
```

Here's a struct that can hold information for one account:

```
typedef struct
{
  int num;
  double bal;
  char name[256];
} Account;
```

Write a **C** program named **p2.c** that reads account information from a file in the above format, storing them in a *dynamically allocated* array of structs, then prints account information exactly as shown below. Your source code should include a comment that shows the command you used to create the executable program.

```
Door: $123.45 in account 4305
Gish: $98.76 in account 1106
Stahl: $1980.28 in account 6816
```

(answer on next page)

5. Add this answer to your file named **answers.txt**. Write a definition for the following function that reads from the indicated stream the account info for one account. The function should return 0 on failure or 1 on success:

```
int readAccountInfo( FILE *fp, Account *ap );
```

```
int readAccountInfo( FILE *fp, Account *ap )
{
  if( fscanf( fp, "%i",  &ap->num  ) != 1 ) return 0;
  if( fscanf( fp, "%lf", &ap->bal  ) != 1 ) return 0;
  if( fscanf( fp, "%s",   ap->name ) != 1 ) return 0;
  return 1;
}
```

```
// p2.c
// gcc -o p2 p2.c

#include <stdio.h>
#include <stdlib.h>

typedef struct
{
  int    num;
  double bal;
  char   name[256];
} Account;

int readAccountInfo( FILE *fp, Account *ap );

int main()
{
  char ifn[] = "data.txt";
  FILE *fp = NULL;

  int i, n;
  Account *acct = NULL;

  fp = fopen( ifn, "r" );
  if( fp == NULL )
  {
    fprintf( stderr, "Couldn't open file %s\n", ifn );
    return 1;
  }

  if( 1 != fscanf( fp, "%i", &n ) )
  {
    fprintf( stderr, "Error reading from %s\n", ifn );
    return 1;
  }

  acct = (Account *)calloc( n, sizeof(Account) );
  if( acct == NULL )
  {
    fprintf( stderr, "Error allocating memory\n" );
    return 1;
  }

  for( i = 0; i < n; ++i )
  {
    if( !readAccountInfo( fp, &acct[i] ) )
    {
      fprintf( stderr, "Error reading account info\n" );
      return 1;
    }

    printf( "%s: $%6.3f in account %i\n", acct[i].name, acct[i].bal, acct[i].num );
  }

  if( acct != NULL ) free( acct );

  return 0;
}

int readAccountInfo( FILE *fp, Account *ap )
{
  if( fscanf( fp, "%i",  &ap->num  ) != 1 ) return 0;
  if( fscanf( fp, "%lf", &ap->bal  ) != 1 ) return 0;
  if( fscanf( fp, "%s",   ap->name ) != 1 ) return 0;
  return 1;
}
```