Answer Key:

1. What is the value of SEEK_CUR? Ans: 1

For items 2-4: A linked list with head pointer L is populated with 3 elements.

2. How many pointers to nodes are found in the list? Ans: 4 (L, node1->link, node2->link, node3->link)

3. How many variables are stored outside of the heap memory? Ans: 1

4. If a pointer A is declared and is currently holding the value of L, how many times does the value of A change if A is used to traverse until it detects the end of the list? Ans: 3

5. Below is a function that checks if a given list is empty or not. It will return Y if it is empty and N otherwise. The datatype of LL is a pointer to a node located in the heap. Fill in the blank to complete the function definition.

```
char checkList(LL *test){
    return (*test == NULL)? 'Y': 'N';
}
```

For items 6 - 8:

Given:

```
typedef struct A{
        long B;
        char C[100];
        float D;
}E;
typedef struct F{
        E G;
        struct F* H;
}*J;

J K;
```

6. Determine the value of sizeof(G) + sizeof(H) + sizeof(J). Ans: 112+8+8 = 128

7. Write the C statement that will allocate for a new node.

Ans:

```
K = (J) malloc(sizeof(struct F));
```

8. Assume that the list is populated with elements, write the C statement that will display the string of the first node of the list.

Ans:

```
printf("%s", K->G.C);
```

Given the code fragments below (which will be performed in order):

```
#define N 5

FILE* fp = fopen("Test2.bin","wb");
int ctr, x, y = sizeof(int), z;

if(fp!=NULL){
    for(ctr=x=0; ctr<N; ctr++){
        x = N % (ctr + 3) + N;
        if(x==5)
            fseek(fp,0,0);
        fwrite(&x,y,1,fp);
    }
    z = ftell(fp);
    fclose(fp);
}
```

9. List in order the entries stored in the file after execution of the code above.

Ans: 5 ~~7~~ ~~6~~ 10 10 or 5 10 10

10. Determine the value of z.

Ans: 3 elements * 4 bytes = 12

Problem A:

```
void insertLateEnrollee(LL* lateEnrollee, student elem)
{
  LL* trav;
  LL temp;

  for(trav = lateEnrollee; *trav != NULL && strcmp((*trav)->S.ID, elem.ID) != 0; trav = &(*trav)->next){}
  if (*trav != NULL){
     printf("Student ID number is taken");
  } else {
     temp = (LL) malloc(sizeof(struct cell));
     temp->S = elem;
     temp->next = *trav;
     *trav = temp;
  }
}
```

Problem B:

```
ComboList createComboList(LL oldList)
{
    ComboList newList = NULL;
    LL travOld;
    LL* travNew;
    LL temp;

    for (travOld = oldList; travOld != NULL; travOld = travOld->next){
      *travNew = (travOld->S.group == 'A') ? &newList.ListA : &newList.ListB;
          for ( ; *travNew != NULL && strcmp((*travNew)->S.ID, travOld->S.ID) < 0; travNew = &(*travNew)->next){
             temp = (LL) malloc(sizeof(struct cell));
             temp->S = travOld->S;
             temp->next = *travNew;
             *travNew = temp;
          }
    }
    return newList;
}
```

Problem C:

```
void writeFinalQualifiers(LL* stud)
{
   LL* trav, temp;
   FILE* fp;

   fp = fopen("FinalsQualifiers.dat", "w");
   if (fp != NULL){
      for (trav = stud; *trav != NULL;){
         if ((*trav)->S.G.SBAscore < 60){
            temp = *trav;
            *trav = temp->next;
            free(temp);
         } else {
            fwrite(&(*trav)->S.N, sizeof(LL), 1, fp);
            *trav = &(*trav)->next;
         }
   }
   fclose(fp);
}
```

Problem D:

```
        int updateRecords()
        {
           FILE* updatedRec;
           student studentRec;
           float finalGrade;
           int passedStud = 0, ctr = 0;

           updatedRec = fopen("StudentGrades.dat", "r+");
           if (updatedRec != NULL){
              while(fread(&studentRec, sizeof(student), 1, updatedRec) != 0){
                 printf("Enter final grade of student# %s: ", studentRec.ID);
                 do {
                   scanf("%.1f",&finalGrade);
                 } while ( !((finalGrade >= 1.0 && finalGrade <= 3.0 ) ||  finalGrade == 5.0);
                 studentRec.G.finalGrade[1] = finalGrade;
                 fseek(updatedRec, ctr*sizeof(student), SEEK_SET);
                 fwrite(&studentRec, sizeof(student), 1, updatedRec);
                 fseek(updatedRec, (ctr+1)*sizeof(student), SEEK_SET);
                 if (finalGrade <= 3.0){
                     passedStud++;
                 }
              }
              fclose(updatedRec);
           }

           return passedStud;
        }
```