ESc101N: Fundamentals of computing(Second mid-semester Exam)

October 13, 2009

Section:
Name:
Roll Number:

Question:	1	2	3	4	5	6	Total
Points:	5	5	5	5	5	5	30
Score:							

Instructions

1. Read these instructions carefully.
2. Write you name, section and roll number on all the pages of the answer book.
3. Write the answers cleanly in the space provided. You can ask for extra sheets for rough work.
4. Using pens (blue/black ink) and not pencils. Do not use red pens for answering.

Section: Name: Rollno:

Question 1. (5 points) What is the output of the following program.

```
#include<stdio.h>
typedef struct{
  int *x;
  int y;
}Foo;

typedef struct{
  int u;
  int *v;
}Bar;

void biz(Foo foo, Bar bar);
int main(){
  int w[] = {15,42};
  Foo foo = {w,w[0]};
  Bar bar = {w[1],w};
  biz(foo,bar);

  printf("w[0]=%d, w[1] = %d\n", w[0],w[1]);
  printf("bar.u = %d\n",bar.u);
  printf("*bar.v = %d\n",*bar.v);
  printf("*foo.x = %d\n",*foo.x);
  printf("foo.y = %d\n", foo.y);


}
void biz(Foo foo, Bar bar)
{
  (*foo.x)++;
  foo.y ++;
  bar.u ++;
  bar.v ++;
}
```

Solution:

```
Script started on Fri 09 Oct 2009 16:11:59 IST
w[0]=16, w[1] = 42
bar.u = 42
*bar.v = 16
*foo.x = 16
foo.y = 15

Script done on Fri 09 Oct 2009 16:11:59 IST
```

Question 2. (5 points) Complete the code given below that implements complex numbers in C

The complete code is given below

```
#include <stdio.h>
typedef double Real;
typedef struct Complex
{
  Real x;
  Real y;
} Complex; /* complex number x + i y */


Complex mul(Complex u, Complex v)
{
  Complex w;
  w.x = u.x * v.x - u.y * v.y;
  w.y = u.x * v.y + u.y * v.x;
  return w;
}

Complex div(Complex u, Complex v)
{
  /* returns u/v  */
  Complex w;
  Real absV;
  absV = v.x * v.x + v.y * v.y;
  w.x  = (u.x * v.x + u.y * v.y) / absV;
  w.y  = (u.y * v.x - u.x * v.y) / absV;
  return w;
}
```

Solution:

1. `w.x  = u.x * v.x - u.y * v.y;`
2. `w.y  = u.y * v.x + u.x * v.y;`
3. `absV = v.x * v.x + v.y * v.y;`
4. `w.x  = (u.x * v.x + u.y * v.y)/absV;`
5. `w.y  = (u.y * v.x - u.x * v.y)/absV;`

Question 3. (5 points) On an unknown machine I ran the program `sizetest.c` given below

```
#include<stdio.h>
int main()
{
   printf("sizeof(int)=%ld, sizeof(int*)=%ld\n", sizeof(int),sizeof(int*));
}
```

and obtained the output `sizeof(int)=4, sizeof(int*)=8`. Predict the output of the following program when run on the same machine.

```
#include<stdio.h>
int main()
{
  int a[100][100], *b[100], (*c)[100];
  printf("%ld\n", sizeof(&a));
  printf("%ld\n", sizeof(a));
  printf("%ld\n", sizeof(*a));
  printf("%ld\n", sizeof(**a));
  printf("%ld\n", sizeof(b));
  printf("%ld\n", sizeof(*b));
  printf("%ld\n", sizeof(**b));
  printf("%ld\n", sizeof(c));
  printf("%ld\n", sizeof(*c));
  printf("%ld\n", sizeof(**c));
}
```

Solution:

```
Script started on Fri 09 Oct 2009 16:13:17 IST
8
40000
400
4
800
8
4
8
400
4
Script done on Fri 09 Oct 2009 16:13:17 IST
```

Question 4. (5 points) A textual screen is captured by the following struct.

```
typedef char *Line;
typedef struct {
  Line *lines; /* lines[i] denotes ith line */
  int nlines;
  int ncols;
} TextScreen;
```

Complete the function `TextScreen * textScreeAlloc(int maxLines, int maxCols)` which allocates using `malloc` enough memory for a text screen with `maxLines` many lines and `maxCols` many columns

The complete code is given below.

```
#include<stdlib.h>
typedef char *Line;
typedef struct {
  Line *lines; /* line[i] denotes ith line */
  int nlines;
  int ncols;
} TextScreen;


TextScreen * textScreenAlloc(int maxLines, int maxCols)
{
  TextScreen *scr;
  if( (scr = (TextScreen *) malloc(sizeof(TextScreen))) == NULL) return NULL;
  if( (scr-> lines = (Line*) malloc(sizeof(Line) * maxLines)) == NULL)
  {
    free(scr); /* free stuff already allocated */
    return NULL;
  }
  for(int i = 0; i < maxLines; i ++)
  {
    if( (scr->lines[i] = (Line)malloc(sizeof(char)*maxCols)) == NULL){
      /* if allocating the ith line failed */
      for(int j = 0; j < i; j ++){
        free(scr-> lines[j]);
      }
      free(scr->lines); /* free the letters field  */
      free(scr); /* free the screen itself */
      return NULL;
    }
  }
  scr -> nlines = maxLines;
  scr -> ncols = maxCols;
  return scr;
}
```

Solution:

1. `(scr = (TextScreen *)malloc(sizeof(TestScreen))) == NULL`
2. `(scr->lines = (Line *)malloc(sizeof(Line) *maxLines)) == NULL`

   The `Lines *` could be replaced by `char **` and `sizeof(Line)` could be replaced by `sizeof(char*)`.
3. `(scr->lines[i] = (Line)malloc(sizeof(char)*maxCols)) == NULL`
4. `free(scr -> lines[j])`
5. `free(scr ->lines)`

Question 5. Assume the following declaration of List data type.

```
typedef struct Cons Cons;
typedef Cons *List;

struct Cons{
  int datum;
  List next;
};
```

(a) (3 points) Complete the definition of the function `void freeList(List)` that frees up the memory used by a given List.

```
void freeList(List a)
{
  List ptr;
  while(a){
    ptr = a;
    a   = a -> next;
    free(ptr);
  }
}
```

Solution:

1. `ptr = a;`
2. `a = a -> next;`
3. `free(ptr);`

(b) (2 points) Assume that the function `List singleton(int x)`, which on input `x` creates a list containing a single element `x`, is already defined. Use this function to complete the following recursive function that makes a copy of a given list.

```
List copyList(List a)
{
  List p;
  if( a == NULL ) return NULL;
  else {
    p = singleton(a->datum);
    p -> next = copyList(a -> next);
  }
  return p;
}
```

Solution:

1. `p = singleton(a -> datum);`
2. `p -> next = copyList(a -> next);`

Question 6. (5 points) Assume again the following definition of linked list.

```
typedef struct Cons Cons;
```

```
typedef Cons *List;

struct Cons{
  int datum;
  List next;
};
```

In the implementation of merge sort with arrays, we used an auxiliary array to merge two sorted arrays. Complete the code given in the *next page* that merges two lists that are sorted according to increasing order. Read the comments in the code carefully. They provide enough hints.

Solution:

1. `return b;`
2. `return a;`
3. `start = a;`
4. `a = a -> next;`
5. `start = b;`
6. `b = b -> next;`
7. `end -> next = a;`
8. `a = a -> next;`
9. `end -> next = b;`
10. `b = b -> next;` Complete solution given in the next page.

```
List merge(List a, List b)
{
  /*
  Function destructively merges two lists. By destructively we mean
  that after the merge the individual lists are destroyed and their
  Conses are rearranged to get the combined list that is sorted.
  */

  /* Handle the case when one of the list is empty */
  if( a == NULL ) return b;
  if( b == NULL ) return a;

  /* Reaches here if and only if both the lists are nonempty. */

  List start; /* The start of the merged list */
  List end; /* The end of the merged list    */

  /* Set up the start of the merged list */
  if( a-> datum < b -> datum){
    start =  a;
    a     = a -> next;
  } else {
    start = b;
    b = b -> next;
  }
  end = start;
  /* Invariant end points to the last element in the merged list so
     far */

  while( a != NULL && b != NULL ){
    if ( a -> datum < b -> datum){
      end ->next  = a;
      a = a -> next;
    } else {
      end -> next = b;
      b = b -> next;
    }
    end = end -> next;
  }

  /* Now at least one of the */

  if( a != NULL) {
    end -> next = a;
  } else {
    end -> next = b;
  }
  return start;

}
```