15-122 : Principles of Imperative Computation

Summer 1 2012

Assignment 7

(Theory Part)

Due: Friday, June 22, 2012 in class

Name: _____

Andrew ID: _____

Recitation: _____

The written portion of this week's homework will give you some practice in transitioning from C0 to C programming basics. You can either type up your solutions or write them *neatly* by hand, and you should submit your work in class on the due date just before lecture begins. Please remember to *staple* your written homework before submission.

Question	Points	Score
1	8	
2	7	
Total:	15	

1. **Programming in C**

   For each of the following problems,

   (a) state what is wrong with the code and

   (b) how to correct it.

   Do not just try to compile it and write down the error message. You can look for errors such as unallocated or uninitialized memory, array index out of bounds, dereference of an invalid pointer, mixing pointer and reference types etc.

(1) (c)

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

int main() {
  char *w;
  strcpy(w,"C programming");
  printf("%s\n", w);
  return 0;
}
```

Solution:

(1) (d)

```
#include <stdio.h>
#define MULT(X,Y) (X*Y)

int main() {
  int c = MULT(2+3,3+4);
  printf("(2+3)*(3+4) is = %d\n", c);
  return 0;
}
```

Solution:

(1) (e)

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

int main() {
  int *a = malloc(100);
  for (int i=0; i<100; i++)
    a[i]=i;
  return 0;
}
```

Solution:

(1) (f)

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

int main() {
  char *name = malloc(strlen("wordpress" + 1));
  strcpy(name,"wordpress");
  return 0;
}
```

Solution:

(1) (g) The standard string library function `strncpy(dest, src, n)` copies the specified number of characters `n` from the source string `src` to the destination string `dest`.

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

int main() {
  char *letter_data = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
  char a[16];
  strncpy(a, letter_data, sizeof(a));
  printf("The first sixteen letters are: %s\n", a);
  return 0;
}
```

Solution:

(1) (h) This code fragment shows a C function that is called from another function.

```
#include "xalloc.h"
#include <stdlib.h>
#define TABLESIZE 100
int *table = NULL;

int insert_in_table(int pos, int value) {
  if (table == NULL)
    table = (int *)xcalloc(TABLESIZE, sizeof(int));
  if (pos >= TABLESIZE)  return -1;
  table[pos] = value;
  return 0;
}
```

Solution:

(1) (i)

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

int main() {
  int a[] = {1,1,2,3,5,8,13,21,34,55};
  printf("%d\n", *(a+2)+*(a+5));
  free(a);
  return 0;
}
```

Solution:

(1) (j)

```
#include <stdio.h>

int main() {
  int a[50];
  int *i;

  for (i = &a[0]; i < &a[51]; i++) {
    *i = 0;
  }
  return 0;
}
```

Solution:

2. **More Programming in C**

Answer the following questions briefly and clearly. Your answers should not be more than few lines of explanation. Provide a simple example when appropriate to support your explanation.

(a) State precisely what the variable `A` points to in this statements

(1) i. `char *A[10];`

Solution:

(1) ii. `char (*A)[10];`

Solution:

(1) iii. `char *(*A)[10];`

Solution:

(b) Which of the following is legal and which one is illegal? Why? Justify your answer.

(1) i.

```
#define banana int
unsigned banana i;
```

Solution:

(1) ii.

```
typedef int banana;
unsigned banana i;
```

Solution:

(2) (c) The first 14 bytes of a bitmap file are reserved for some header information that is defined by the following struct.

```
struct bmp_header
{
  unsigned short int type; /* BMP type identifier */
  unsigned int size;    /* size of the image file in bytes*/
  unsigned short int reserved1, reserved2;
  unsigned int offset;   /* starting address of the byte */
};
```

Assume that `unsigned short` values are 2 bytes and `unsigned int` values are 4 bytes. Suppose we read the first 14 bytes of a bitmap file into a header that is defined as below.

```
char header[14];
```

Assuming that struct fields are laid out contiguously in memory, how would you extract the size of the image from the given data? Write a line of code that will find the size of the image in bytes.

Solution: