

Name _____

Section _____

CS2303 C08
Systems Programming Concepts
Mid Term Exam
February 1, 2008
Answers A and B Versions

Question	Points	Score
0	1	
1	5	
2	4	
3	5	
4	20	
5	15	
6	18	
7	16	
Total	80	

Trivia Question (1 extra credit point)

0. (a) Where will the Superbowl be played in 2009?

Tampa, Florida

-OR-

(b) Who was the last American Vice President to be elected President?

George H. W. Bush

1. Given the following screen output from a ccc computer and assume you are fred:

```
$ pwd
/home/fred/q1
$ cd qlow
$ ls -la
total 8
drwxr-x--- 2 fred 6810 4096 Jan 30 14:52 .
drwxr-x--- 3 fred 6810 4096 Jan 30 14:49 ..
-rwx----- 1 fred 6810  12 Jan 30 14:37 file1.c
-rwx----- 1 fred 6810  34 Jan 30 14:38 fileB.c
```

(4 pts) a. Give the necessary command lines to copy file1.c to /home/fred/q1/fileA.c.

```
cp file1.c ../fileA.c
```

(1 pt) b. Which file is larger fileB.c or file1.c?

```
fileB.c
```

(2 pts) 2a. Explain how a pointer gets typed.

It is typed based on the type of variable or structure it points to. This is specified in the declaration.

(2 pts) 2b. How does the type of a pointer affect pointer arithmetic?

In pointer arithmetic such as `ptr + 1` the plus one increases the address held in the pointer by the `sizeof` in integer bytes assigned to the type.

(2 pts) 3a. Name the two most important concerns when evaluating computer system.

```
efficiency and fairness
```

(3 pts) 3b. Using a queuing system model define the mean response time of n customers in a cpu scheduler.

In a queuing system a customer arrives at time t_A and leaves the system at time t_B . The i th customer's response is $t_{Bi} - t_{Ai}$. Thus the mean response time is $\text{Sum}(i\text{th response time})/n$. Response time includes time in queue + time in service.

(20 pts) 4. What is the output from this program?

```

int gl = 4;
float zippy (float y, int i)
{
    static z = 5.0;
    --z;
    return 2*i + y - z;
}
int mix( int i, int v[5], float j)
{
    int t[5] = {3, 6, 9};
    t[i] = (int)j - i;
    gl = gl*gl;
    v[i] = 10*gl;
    j = i + 3.0;
    i++;
    t[i+1]++;
    printf("A : %d %6.2f %d %d\n", i, j, gl, t[i]);
    return t[i];
}
int main ()
{
    int i=1, j, k;
    int t[5] = {50,60,70,80};
    float x, y, z;

    x = (float)i/2;
    y = 8.0*i;
    z = y/5;
    j = 20;
    printf("%6.2f %6.2f %6.2f\n", x, y, z);
    for (k = 0; k < 2; k++)
    {
        gl = mix(i,t,(float)j);
        j = zippy(z,t[i]);
        printf("Main: %d %d %d %d\n",
            i, j, gl, t[i]);
        i++;
    }
    printf("T:");
    for (j = 0; j < 5; j++)
        printf(" %d", t[j]);
    return 0;
}

```

Output Box B

Output Box A

0.50	8.00	1.60	[6]	0.50	8.00	1.60	[3]				
B :	2	4.00	16	9	A :	2	4.00	16	9	[4]	
Main:	317	20	9	orange	Main:	1	317	9	160	[4]	
					A :	3	5.00	81	0	[4]	
					Main:	2	1618	0	810	[2]	
					T:	50	160	810	80	0	[3]

(15 pts) 5. What is the output from this program?

```

#define SIZE 3
int main ()
{
  int i, y[SIZE] = {100,300,500};
  int x[SIZE] = {4000};
  int *Sptr, *Fptr;

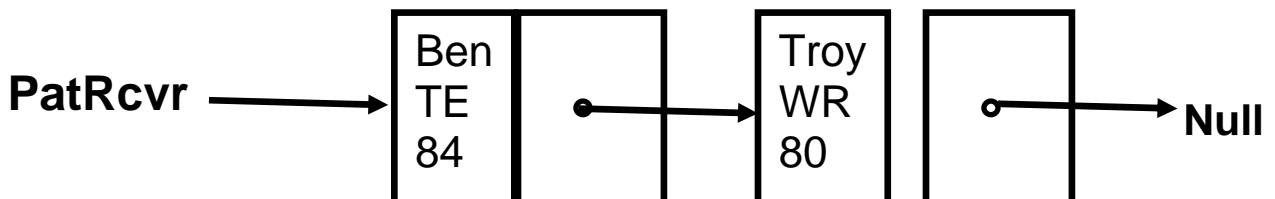
  Fptr = (int *) malloc(sizeof (i));
  Sptr = (int *) malloc(sizeof (i));
  Fptr = x;
  for (i=0; i < SIZE; i++)
  {
    *Fptr = y[2-i] + 20*i;
    Fptr++;
    printf("i = %d y = %d x = %d\n", i, y[i], x[i]);
  }
  *Sptr = *(--Fptr);
  printf ("%d %d %d\n", y[i-1],*(Fptr-1),*Sptr);
  Sptr = (int *) malloc(sizeof (i));
  printf ("%d %d\n", *Sptr, *Fptr);
  *Sptr = *(Fptr-2);
  Fptr = (int *) malloc(sizeof (i));
  free(Fptr);
  printf ("%d %d\n", *Sptr, *Fptr);
  return 0;
}

```

Output Box A

Output Box B

i = 0 y = 100 x = 500	i = 0 y = 100 x = 600	0.5 0.5 1 = 2
i = 1 y = 300 x = 320	i = 1 y = 300 x = 320	0.5 0.5 1 = 2
i = 2 y = 500 x = 140	i = 2 y = 600 x = 140	0.5 0.5 1 = 2
500 320 140	500 320 140	1 1 1 = 3
unk 140	unk 140	1.5 1.5 = 3
500 unk	600 unk	1.5 1.5 = 3



(16 pts) 6a. Assume the linked list **PatRcvr** currently looks like the figure above. Write inline code (not a separate fcn) that assumes that **PatRcvr** is ordered by number and not NULL to insert a new node with contents (Randy, 81, WR) into **PatRcvr** in the correct position in the linked list.

(2 pts) Redraw the linked list to show its contents after this insertion.

```
#include <stdio.h>
typedef struct{
    char *name;
    char *position;
    int number;
    struct Player *pptr;
} Player;
typedef Player *Pptr;
const char *names[] = {"Tom", "Troy", "Randy", "Ellis", "Ben", "Wes"};
const int nums[6] = {12, 80, 81, 27, 84, 83};
const char *pos[] = {"QB", "WR", "WR", "CB", "TE", "WR"};

Pptr createnode (int index)
{
    Pptr newPtr;

    newPtr = malloc(sizeof(Player));

    newPtr->name = names[index];
    newPtr->position = pos[index];
    newPtr->number = nums[index];
    newPtr->pptr = NULL;
    return newPtr;
}
int main()
{
    Pptr PatRcvr = NULL; Pptr previousPtr = NULL;
    Pptr RcvPtr, currentPtr ;
    int i;

    for (i=1; i < 6; i = i+3)
    {
        RcvPtr = createnode (i);
        RcvPtr->pptr = PatRcvr;
        PatRcvr = RcvPtr;
    }
    ... /* PatRcvr has two nodes on list as above */
```

```

RcvPtr = createnode (2);           [4] RcvPtr = createnode (2);
currentPtr = PatRcvr;             [6] RcvPtr-> pptr = PatRcvr -> pptr;
                                   [6] PatRcvr->pptr = RcvPtr;

/* loop to find correct location in list */
while ( currentPtr != NULL
        && currentPtr->number > RcvPtr-> number)
{
    previousPtr = currentPtr;
    currentPtr = currentPtr->pptr;
} /* end while */

/* insert at beginning of list */
if ( previousPtr == NULL )
{
    RcvPtr->pptr = PatRcvr;
    PatRcvr = RcvPtr;
} /* end if */

/* insert node between previousPtr and currentPtr */
else
{
    previousPtr->pptr = RcvPtr;
    RcvPtr->pptr = currentPtr;
} /* end else */
return;
}

```

Before:

Pat Rcvr	Ben TE 84	Troy WR 80	NULL
----------	-----------	------------	------

After:

PatRcvr	Ben TE 84	Randy WR 81	Troy WR 80	NULL
---------	-----------	-------------	------------	------

(16 pts) 7. Smashball has been expanded to allow players to move in 8 directions

Write a simple function **diag_NE** to test movement in the Northeast direction. This is the function prototype:

```
diag_NE ( char figure[SIZE ] [SIZE], char schar, int row, int col);
```

After filling the figure with blanks, **diag_NE**, puts **schar** into location **row**, **col** in the figure. The function continues by moving in the Northeast direction (one row up, one column right) depositing **schar** in the new spot until it has reached an edge (top or right). After depositing **schar** on the edge, the function returns.

As an example assume the call to the function is inside this code snippet:

```
#define SIZE 40
```

```
#define SIZE1 SIZE-1
```

```
enum direction {E, NE, N, NW, W, SW, S, SE};
```

```
int BLANK = ' ';
```

```
int main {
```

```
char field [SIZE][SIZE]
```

```
dir = ...
```

```
switch (dir)
```

```
    case NE:
```

```
        diag_NE( field, 'D', 20, 18 );
```

```
        break;
```

```
}
```

```
int diag_NE (char figure[SIZE][SIZE], char schar, int row, int col)
```

```
{
```

```
    int i,j;
```

```
    for (i=0; i < SIZE; i++)
```

```
        for (j=0; j < SIZE; j++)
```

```
            figure[i][j] = BLANK;
```

```
    while (row != 0 && col != SIZE1)
```

```
[3] + [3]
```

```
    {
```

```
        figure[row][col] =schar;
```

```
[3]
```

```
        row--;
```

```
[3]
```

```
        col++;
```

```
[3]
```

```
    }
```

```
    figure [row][col] = schar;
```

```
[1] end condition correct
```

```
}
```

```
[-1] putting Blank back in
```

A: This is an intentional Blank page to show your work!