Name: _____

In-Class, Open Book Final Examination
May 18, 2012

The work on this examination is to be your own and you are expected to adhere to the UMass-Boston honor system. All questions can be answered by one or two short sentences. Do not try to make up for a lack of understanding by providing a rambling answer.
Note: I give partial credit!   Show all work!

1. (20 points)
Circle True(T) or False(F) for each question. If the question is not clear, write in your interpretation.

T or F	*(pa +1) is the same as *pa++
T or F	The compiler will flag an error with the following: struct name{unsigned int data : 3;} flag; flag.data =8;
T or F	You need to understand Newton's Third Law in order to do hw7.
T or F	The program will hang at scanf() if you try to run it in the background mode.
T or F	struct name {int a;}*p; The member of the struct is obtained by p.a
T or F	b = = (b^a^b);
T or F	Function pointer is used to pass different function names to a function.
T or F	The word junk will get print out from the statement: system("echo junk");
T or F	The following statement is a definition: union junk{int a; char *b;}abc;

2. (20 points)
a. (10 points) Fill in the values. You can write &… for an address. If they are unknown , enter "NA". Assume the pointers get adjusted after evaluation.

```
typedef struct {
      int   i;
      char  *str;} nstring;

nstring n[] ={{12, "Dec"},{2, "Feb"},{6, "Jun"},
            {1, "Jan"}, {10, "Oct"}};
nstring *np1, *np2;

np1 = n + n[1].i;/*points where? _____*/

np2 = n + (n + 3)->i;/*points where? _____*/

printf("%d, %s\n", np1->i, np2->str);

      /*What prints? _____*/
printf("%d, %s\n", --np1->i, ++np2->str);

      /*What prints? _____*/
printf("%d, %s\n", (--np1)->i, (++np2)->str);

      /*What prints? _____*/
```

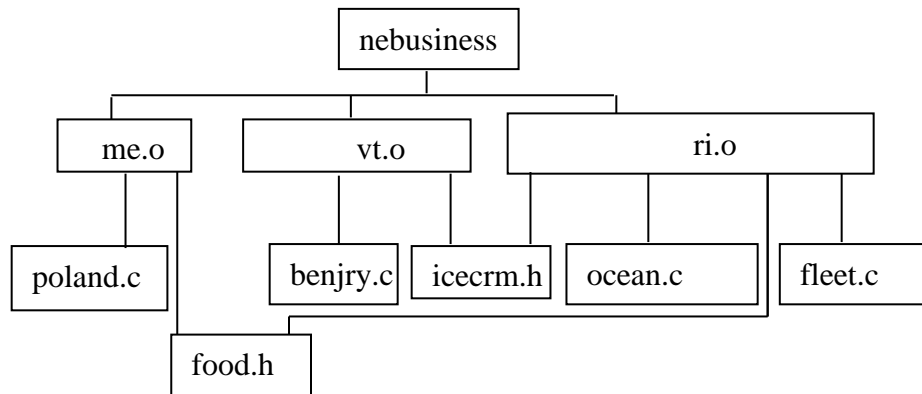b. (10 Points) The program for grep.c is as follows:

```
    #include <stdio.h>
    int main(int argc, char *argv[]){
     printf("%x,\n", argc);
     printf("%x,\n",*argv[1]);
     printf("%x,\n", (*++argv)[0]);
     printf("%x,\n",*++argv[0]);return 0;}
```

What will print out if I invoke the UNIX command?

  % ./grep –i test typescript

_____   _____   _____   _____

argc   *argv[1] (*++argv)[0] *++argv[0]

3. (20 points) makefile

```
                        ┌──────────────┐
                        │  nebusiness  │
                        └──────────────┘
        ┌───────────────────┼───────────────────────┐
   ┌─────────┐      ┌──────────────┐      ┌─────────────────────┐
   │  me.o   │      │     vt.o     │      │        ri.o         │
   └─────────┘      └──────────────┘      └─────────────────────┘
        │         ┌──────┴──────────┐    ┌──────┬───────────┬──────────┐
  ┌───────────┐ ┌──────────┐ ┌───────────┐ ┌──────────┐ ┌──────────┐
  │  poland.c │ │ benjry.c │ │ icecrm.h  │ │ ocean.c  │ │ fleet.c  │
  └───────────┘ └──────────┘ └───────────┘ └──────────┘ └──────────┘
        │     ┌──────┘
   ┌───────────┐
   │  food.h   │
   └───────────┘
```

The dependency list of making nebusiness is shown in the above diagram.

i) (10 points) Write a makefile to make nebusiness. Include a clean option to delete all object files.

ii)   (5 points) If you modify the file food.h, and re-make nebusiness, write down what make will do.

iii)   (5 points) If you want to substitute the header file icecrm.h with business.h during compilation time without changing the make file, write one make file that can accommodate this.

4.(40 points) Programming Question:
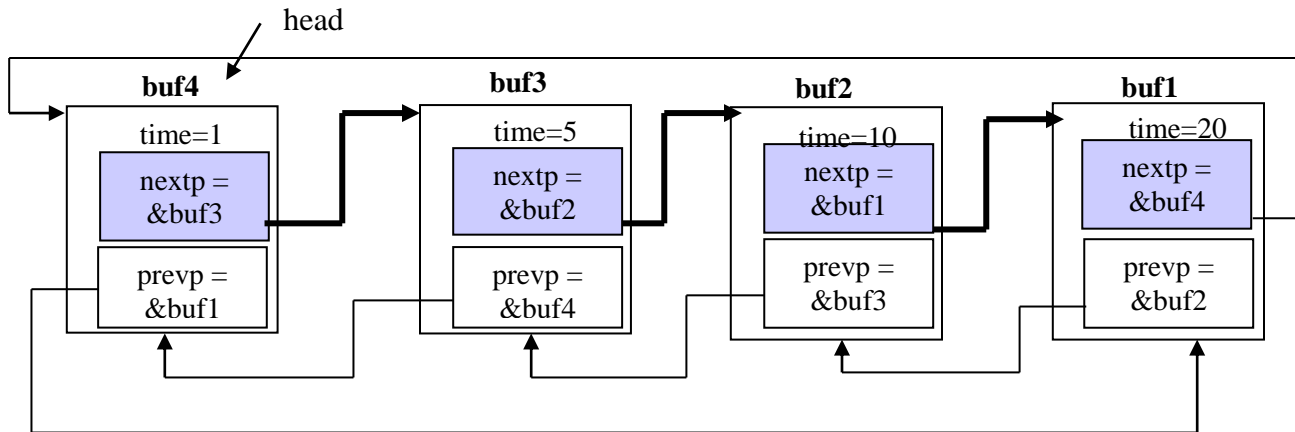
head

A priority queue structure is constructed using a
linked list whose structure is defined as:

```
struct link{
 int time;
 struct link * nextp;
 struct link * prevp;
};
struct link buf,buf1, buf2, buf3, buf4;
struct link * head_ptr= &buf4;
main(){
…
enque(head_ptr,  &buf);
…}
```

The order of the linked list is based on
time. An example is shown in the above diagram.
Write the enque() function that will queue the struct
buf to the linked list according to increasing time.
Show your pseudo code and C code.

```
static void enque(struct link * head, struct link *buf)
{…}
```

**For your reference, a copy of enchain
function from HW7 is provided:**

```
static void enchain(struct blockl *p)
{
    struct blockl *holdp;/* temporary holder*/
    holdp = freep;      /* remember old block
                          at head of chain */
    freep = p;          /* place new block
                          on head of chain */

     /* update structures to place new block
        into chain    */
    if (holdp == NULL)  { /* free chain had
                          become empty */
      freep->nextp = freep;/* one block,
                          points to self */
      freep->prevp = freep; /* ditto */
      cursorp = freep;      /* and cursor pts
                          to this blk   */
    }
    else  {       /* old free chain non-empty*/
              /* insert on nextp list-- */
      freep->nextp = holdp; /* new block
                          points to old block */
      holdp->prevp->nextp = freep;
          /* end block points to new block*/
          /* and insert on prevp list--*/
      freep->prevp = holdp->prevp;
          /* new block points to end block*/
      holdp->prevp = freep;
          /* old block points to new block*/
    }
}
```

Answers:

1. F　　T　　F　　T　　F　　　F　　T　　T　　T

2 a)

&n[2]
&n[1]
6, Feb
5, eb
2, Jun

b)

blade64(2)% ./grep –i test typescript
4　　0x2d or '-'　　0x2d or '-'　　0x69 or 'i'

3
i)
nebusiness: me.o vt.o ri.o
　　　gcc me.o vt.o ri.o –o nebusiness

me.o: poland.c food.h
　　　gcc –c –o me.o poland.c

vt.o: benjry.c icecrm.h
　　　gcc –c –o vt.o benjry.c

ri.o: icecrm.h ocean.c food.h fleet.c
　　　gcc –c –o ri.o ocean.c fleet.c

clean:
　　　rm *.o

ii)
gcc –c –o ri.o ocean.c fleet.c
gcc –c –o me.o poland.c
gcc vt.o me.o　 ri.o –o nebusiness

iii)
Use macro definition and rewrite the makefile as:

HDR=icecrm.h
nebusiness: me.o vt.o ri.o
　　　gcc me.o vt.o ri.o –o nebusiness

me.o: poland.c food.h
　　　gcc –c –o me.o poland.c

vt.o: benjry.c $(HDR)
　　　　gcc –c–o vt.o benjry.c

ri.o: $(HDR) ocean.c food.h fleet.c
　　　　gcc –c–o ri.o ocean.c fleet.c

clean: rm *.o

To build new version with business.h, type:
　　make HDR=business.h

To make the old version with icecrm.h, type:
　　make

4.
```c
/* Program to do priority Queue */
#include <stdio.h>
struct  link{
  int time;
  struct link * nextp;
  struct link * prevp;
};

void enqueue(struct link *, struct link *);

main()
{
  struct link buf1, buf2,buf3, buf4;
  struct link *head_ptr = &buf4, *now1,
*temp1=NULL;

/* initialize the first struct buf4 */
  buf4.time =1;
  buf4.nextp =&buf4 ;
  buf4.prevp = &buf4;

  for(now1= head_ptr; now1 !=temp1; now1=
  now1->nextp){
   temp1=head_ptr;
   printf("buf =%p  buf->time= %d, buf->nextp=%p,
buf->prevp=%p\n",now1, now1->time, now1->nextp,
now1->prevp);
  }
```

```
  printf("\n");

/* add 1st struct buf3 with buf3.time=5 */
  buf3.time =5;
  buf3.nextp=NULL;
  buf3.prevp=NULL;

  enqueue(head_ptr,&buf3);
  temp1=NULL;
  for(now1= head_ptr; now1 !=temp1; now1=now1-
>nextp){
   temp1=head_ptr;
  printf("buf =%p  buf->time= %d, buf->nextp=%p,
buf->prevp=%p\n",
       now1, now1->time, now1->nextp, now1-
>prevp);
  }
  printf("\n");


/* add 2nd struct buf1 with buf1.time=20 */
 buf1.time =20;
 buf1.nextp=NULL;
 buf1.prevp=NULL;

 enqueue(head_ptr,&buf1);
 temp1 = NULL;
  for(now1= head_ptr; now1 !=temp1; now1=now1-
>nextp){
   temp1=head_ptr;
  printf("buf =%p  buf->time= %d, buf->nextp=%p,
buf->prevp=%p\n",
       now1, now1->time, now1->nextp, now1-
>prevp);
  }
  printf("\n");

/* add 3rd struct buf1 with buf2.time=10 */
  buf2.time =10;
  buf2.nextp=NULL;
  buf2.prevp=NULL;

  enqueue(head_ptr,&buf2);
  temp1 = NULL;
  for(now1= head_ptr; now1 !=temp1; now1=
now1->nextp){
   temp1=head_ptr;
```

```
   printf("buf =%p  buf->time= %d, buf->nextp=%p,
buf->prevp=%p\n",now1, now1->time, now1->nextp,
now1->prevp);
  }
  printf("\n");

}


/* enqueue function according to time */
void enqueue(struct link * head, struct link *buf)
{
  struct link *now, *temp=NULL;

/* compare with the linked list members and if value
time is less, go to another member */

  for(now= head; now !=temp; now=now->nextp){
    temp=head;
    if(now->time > buf->time)break;
  }

  buf->nextp= now;
  buf->prevp= now->prevp;
  now->prevp = buf;
  buf->prevp->nextp = buf;

 return;
}
```