Midterm Review

CS 211

Fall 2018

Midterm – Friday 10/26/18 during Lecture

- BSB 250
- Official Time: 10:00 am to 10:50 am
  - If we can start a few minutes early, we will
  - So try to arrive closer to 9:50am
  - We will need to finish right at 10:50 since there is a class right after ours
  - If you arrive late, you lose time
    (been shown to have negative impact on your final score)

- This is an “Open Brain” Exam
  - No notes, texts, calculators, friends, etc.

Midterm – Friday 10/26/18 during Lecture

- About 4 Questions
- You will be writing C code
- More emphasis on Sematics (meaning) than Syntax
  - But proper syntax is still important and expected
- Think about important code pieces from Projects and Labs

Midterm – Friday 10/26/18 during Lecture

- C Pointers is the Main Topic
  - Dynamic Arrays
  - Linked Lists
- Use of malloc() is expected/required!
- Use of C Pointers in C structs with typedefs
- Assume variables are declared in main( )
- Assume variables are passed to functions
  - Pass-by-value and pass-by-reference knowledge is expected
- Assume manipulation occurs in these functions

Midterm – Friday 10/26/18 during Lecture

- Dynamic Arrays

```
typedef struct dynarrStruct
{
    int*    dynArray;
    int     allocated;
    int     inUse;
} dynarr;
```

- Operations
  - initialize
  - insert value (and growing array)
  - access and use of structure

Midterm – Friday 10/26/18 during Lecture

- Dynamic Arrays declared in main( )

```
main( ) {
    dynarr arr1;     /* create instance on the stack */

    init ( &arr1);    /*call initialization function */

    addValue ( &arr1, val );          /* insert value */

    if ( isEmpty( arr1 ) == TRUE )   /* use/access of array */
            …
    …
}
```

Midterm – Friday 10/26/18 during Lecture

- Dynamic Array instance initialized to be able to store 10 values:

```
void init ( dynarr & arr) {   /* Note: Pass-by-Reference Param */
    arr->allocated = 10;

    arr->dynArray = ( int* ) malloc ( sizeof(int) * arr->allocated);

    arr->inUse = 0;
}
```

Midterm – Friday 10/26/18 during Lecture

- Value added to Dynamic Array instance :

```
void addValue ( dynarr & arr, int val) {      /* Note: Pass-by-Reference param */
        if ( arr->inUse >= arr->allocated ) {
                int* tmp = arr->dynArray
                arr->dynArray = ( int* ) malloc ( sizeof(int) * (arr->allocated * 2) );
                int a;
                for ( a = 0 ; a < arr->inUse ; a++)
                        arr->dynArray[a] = tmp[a];
                free (tmp);
                arr->allocated = arr->allocated * 2;
                }
        arr->dynArray[arr->inUse] = val;
        arr->inUse = arr->inUse + 1;
}
```

Midterm – Friday 10/26/18 during Lecture

- Dynamic Array other functions :

```
int isEmpty ( dynarr arr ) {
        if ( arr.inUse == 0)                    /* using . since pass-by-value param */
                return TRUE;
        else
                return FALSE;
    }

int getLastValue( dynarr arr ) {
        if (isEmpty (arr) == TRUE )
                return -999;   /* some error value */
        else
                return arr.dynArr [ arr.inUse – 1 ] ;
    }
```

Midterm – Friday 10/26/18 during Lecture

- Dynamic Array other functions :

```
int removeLastValue( dynarr* arr ) {        /* NOTE: Pass-by-Reference param */
    if (isEmpty (*arr) == TRUE )
            return FALSE;   /* some error value */
    else
            arr->inUse =  arr->inUse – 1 ;
  }

int getNthValue( dynarr arr , int n) {        /* NOTE: Pass-by-Value param */
    if ( (isEmpty (arr) == TRUE ) || (n >= arr.inUse ) )
            return -999;   /* some error value */
    else
            return arr.dynArr [ arr.inUse – 1 ];
  }
```

Midterm – Friday 10/26/18 during Lecture

- Linked List

```
typedef struct nodeStruct
{
    int         elem;
    struct nodeStruct*      next;
} node;
```

- Operations
  - initialize
  - insert value
  - access and use of list
  - remove value

Midterm – Friday 10/26/18 during Lecture

- Linked List declared in main( )

```
main( ) {
    node* head;      /* create pointer on the stack */

    /*initialize list to empty – multiple approaches */
    init ( &head);
    head = init2 ( );
    head = NULL;

    addToList ( &head, val );                    /* pass-by-reference: modifying list */
    removeValueFromList ( &head, val );
    removeFirstNode (&head );
    val = getNthValue ( head, position );        /* pass-by-value: accessing list */
    length = listLength ( head);
```

Midterm – Friday 10/26/18 during Lecture

- Linked List initialization

```
void init (node** hd) {
        *hd = NULL;
}

node* init2 ( ) {
        return NULL;
}
```

Midterm – Friday 10/26/18 during Lecture

- Linked List add value
- See Code on Lab Pages for examples (lab6.c has them all)

- malloc ( ) statement

  node* tmp = (node*) malloc (sizeof (node));

- Common Insertion Points
  - Insert at the beginning of the list
  - Insert at the end of the list
  - Insert in increasing order

Midterm – Friday 10/26/18 during Lecture

```
void insertInOrder (node** hd, int val) {
      node* curr = *hd;
      node* prev = NULL;

     while ((curr != NULL) && (curr->elem < val))   {        /* set curr to node in the list that is >= val */
        prev = curr;                                          /*   set prev to node just before curr */
        curr = curr->next;
       }

     node* ptr = (node*) malloc (sizeof(node));              /* create the node to add into the list */
      ptr->elem = val;
      ptr->next = curr;

     if (prev == NULL)                                       /* if prev is null, insert at the front of the list */
            *hd = ptr;
     else
           prev->next = ptr;                                 /* otherwise insert right after prev */
     }
```

Midterm – Friday 10/26/18 during Lecture

- Linked List Usage – determining length

```
int listLength (node* hd)   {
 int length = 0;
 while (hd != NULL)    {
    length++;
    hd = hd->next;
   }
 return length;
}
```

Midterm – Friday 10/26/18 during Lecture

- Linked List Usage – print values in the list

```
int show (node* hd)   {
        while (hd != NULL)    {
                printf (“%d, “, hd->elem );
                hd = hd->next;
                }
        printf (“\n”);
}
```

Midterm – Friday 10/26/18 during Lecture

- Linked List Usage – get the Nth vlaue

```
int show (node* hd, int n)   {
        int position = 0;
        while (hd != NULL)    {
                if ( position == n )
                        return hd->elem;
                position++;
                hd = hd->next;
                }
        return -999; /* some error value because list is too short */
}
```