**Computer Science 2211b Final Examination**

**xx April 20xx**
**3 hours**

Student Number: _____

Surname: Given name:

**Instructions/Notes**: The examination has **40 questions** on **15 pages,** and a total of **150 marks**. Put all answers on the question paper. (Circle your multiple choice and true/false answers.)

This is a closed book exam**. NO ELECTRONIC DEVICES OF ANY KIND ARE ALLOWED**. Students are allowed to bring one 8.5x11 sheet of notes.

1. [2 marks] consider a C program that contains the declarations:

   **int k, *p;**

   Either of the following statements could be used as the next line of the program to successfully read in an **int** value:

   **scanf("%d", &k);**
   **scanf("%d", p);**

   a. True
   b. False

2. [2 marks] The following is a portion of a C program:

   ```
   int x=4, y=6, *p;
   p = &x;
    (*p) = y;
   p = &y;
    (*p) = x;
   ```

   What are the values of **x** and **y** after the last statement?

   a. x=4, y=4
   b. x=4, y=6
   c. x=6, y=4
   d. x=6, y=6
   e. None of the above

3. [1 mark] When **printf( )** is used to print out the contents of a string, it uses the function **strlen( )** from <string.h> to determine how many characters to print.

   a. True
   b. False

4. [1 mark] Suppose that a C program is reading from an input file. End of file is detected only when the program makes an attempt to read beyond the end.

   a. True
   b. False

For questions 5 through 15, suppose that a C program contains the following type definitions, function prototypes, and variable declarations. Also suppose that variables **t** and **p** have been initialized to (nonempty) **Things**.

```
typedef     struct thing {
                char   *who;
                int    marks[10];
                char   code;
        } Thing;

void fnOne( int k, int *iptr);
void fnTwo( char c, char *cptr);

Thing t, *p;
```

5. [2 marks] The function call **fnOne(t.marks[2], p->marks[2]);** is syntactically correct.

    a. True
    b. False

6. [2 marks] The function call **fnOne((*p).marks[2], &(t.marks[2]));** is syntactically correct.

    a. True
    b. False

7. [2 marks] The function call **fnOne(617, (*p).marks);** is syntactically correct.

    a. True
    b. False

8. [2 marks] The function call **fnOne(int(p->code), &((*p).marks[2]));** is syntactically correct.

    a. True
    b. False

9. [2 marks] The function call **fnOne( t->marks, p->marks);** is syntactically correct.

    a. True
    b. False

10. [2 marks] The function call **fnTwo( t.code, p->who);** is syntactically correct.

    a. True
    b. False

11. [2 marks] The function call **fnTwo( t.who, p->who);** is syntactically correct.

    a. True
    b. False

12. [2 marks] The function call **fnTwo( t.who[0], p->who[0]);** is syntactically correct.

    a. True
    b. False

13. [2 marks] The function call **fnTwo( *((*p).who), &(p->who));** is syntactically correct.

    a. True
    b. False

14. [2 marks] The function call **fnTwo( *(t.who), &(t.code));** is syntactically correct.

    a. True
    b. False

15. [2 marks] The function call **fnTwo( p->who->, &(p->code));** is syntactically correct.

    a. True
    b. False

16. [1 mark] The declaration

    **int arr[ ] = {2, 4, 6, 8, 10};**

    creates and initializes a statically allocated array that holds 5 elements.

    a. True
    b. False

17. [1 mark] The declaration

    **int b[5][4];**

    creates a two-dimensional array that can store 5 rows of integers, with 4 integers in each row.

    a. True
    b. False

18. [2 marks] The C code segment

```
int j, k, a[3][3];
for (k=0, j=0; k<3, j<3; k++, j++)
   a[k][j] = 5;
```

    initializes all elements of array **a** to the value 5.

    a. True
    b. False

19. [2 marks] Suppose that a C program is attempting to execute the statement

    **scanf("%d", &k);**

    where k is a variable of type int, and the input entered by the user is

    **abcde**

    A run-time error occurs at this point, and the program crashes.

    a. True
    b. False

20. [1 mark] The only items in a C program that can be deallocated using the standard library function **free( )** are ones that were created dynamically.

    a. True
    b. False

21. [1 mark] In C, the *null character* is another name for the *null pointer*.

    a. True
    b. False

22. [1 mark] All parameters to C functions are passed *by reference*.

    a. True
    b. False

23. [1 mark] To convert an **int** value to a string value, a C program can use the function **sprintf( ).**

    a. True
    b. False

24. [1 mark] The C function **fclose( )** is used to delete files.

    a. True
    b. False

25. [2 marks] Consider a C program containing the declarations

```
double arr[ ] = {3.2, 6.1, 9.5};
double *p = arr;
```

    Executing the statement

```
++p;
```

    will increment the value stored in **p** by **sizeof(double)**.

    a. True
    b. False

26. [1 mark] All variables inside a shell script program are stored as strings.

    a. True
    b. False

27. [3 marks] Complete the following statement:

*A C function cannot have a pointer to a local variable in the function as its return value because*

*Local* variables (variables with *block scope*) are only accessible within the block of code in which they are declared. Therefore such pointer will point to a memory location that is not available after the function call completed.

28. [15 marks] Consider a program in which all the **.c** and **.h** files are in the same directory as the following **makefile**:

```
main: main.o api1.o api2.o api3.o api4.o
        gcc –o main main.o api1.o api2.o api3.o api4.o

main.o: main.c *.h
        gcc –c main.c

api4.o: api4.c
        gcc –c api4.c

api3.o: api3.c a.h e.h f.h
        gcc –c api3.c

api2.o: api2.c c.h d.h e.h f.h
        gcc –c api2.c

api1.o: api1.c a.h b.h c.h d.h
        gcc –c api1.c

clean:
        rm –f  *.o  main
```

a. List, in order, the compilation commands that will be carried out when the user enters the command sequence

**make clean; make main**

```
rm –f *.o main
gcc –c main.c
gcc –c api1.c
gcc –c api2.c
gcc –c api3.c
gcc –c api4.c
gcc –o main main.o api1.o api2.o api3.o api4.o
```

For this program, the commands **make** and **make main** will always do the same thing.

i. True
ii. False

b. Suppose that, since the last time that the executable was built, files **api4.c** and **b.h** have been edited. List, in order, the compilation commands that will be carried out when the user enters the command

**make main**

gcc –c main.c
gcc –c api1.c
gcc –c api4.c
gcc –o main main.o api1.o api2.o api3.o api4.o

c. State what the **–I** command line option for **gcc** is used for, and explain why it is not needed in this **makefile**.

The -I option is for cc and gcc to specify a path (or paths) on which to look for .h files that are mentioned in statements of the form #include "StackTypes.h" in .c files.

29. [6 marks] A correctly-working C program contains a declaration for the structured type **record**, and includes the following statements (not on consecutive lines) somewhere in the main program:

```
record  recA, *recB;
*(recA.mem1) = 19.45;
printf("%s", recA.mem2);
free(recA.mem2);
recA.mem3[6] = 8;
recB->mem4 = &recA;
recB->mem5 = recA.mem3[0];
```

Provide a type definition for **record** that is consistent with these statements.

```
struct Rec {
  float mem1[2];
  char *mem2;
  int mem3[10];
  struct Rec *mem4;
  int mem5;
};
typedef struct Rec recond;
```

30. [8 marks] A C main program for an unspecified application begins by accepting a single command line argument, and attempting to open an output file named by the argument. If the user has supplied the incorrect number of parameters, or if the file cannot be opened, the program issues an error message and terminates. Write the portion of the program that carries out this task. Be sure to include the header for the main program, any necessary **#include** statements, etc, in your answer.

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
  FILE *fp;
  if (argc!=2) {
    printf("Error: incorrect number of argument\n");
    exit(1);
  } else if ( (fp=fopen(argv[1], "r"))==NULL ) {
    printf("Error: cannot open file: %s\n", argv[1]);
    exit(2);
  }
  else {
    ;
  }
}
```

Questions 31 through 34 use the type definitions for structured types **card**, **cardNode**, **cardList**, **cardQueue** and **cardStack** found in the reference notes that accompany this exam. (These are the same ones that were used on Assignment 6.)

31. [8 marks] Complete the following code segment so that the card **c** holds the queen of clubs, and is added to the front of each linked list.

```
card  c;
cardList list1;
cardList * list2;
```

32. [9 marks] Complete the following function so that it reverses the order of the elements in the linked list referenced by **list**. (Assume that you have access to all the structure modules from Assignment 6.)

```
void reverse( cardList * list )
```

33. [6 marks] Write a function **deallocateCardList**, for inclusion in cardListApi.c, that deallocates the entire contents of the linked list referenced by **list**.

    **void deallocateCardList( cardList * list )**

34. [4 marks] Write a function **deallocateCardQueue**, for inclusion in cardQueueApi.c, that deallocates the entire contents of the queue referenced by **cq**. You may make use of the function defined in the previous question.

    **void deallocateCardQueue( cardQueue * cq )**

35. [6 marks] Write a C function called **myStrchr** that performs the same operation as the function **strchr** from <string.h>. That is, **myStrchr** returns a pointer to the first occurrence of character **ch** in the string referenced by **s**; if **ch** is not found in the string, the function returns the null pointer instead. You are ***not allowed*** to use any functions from <string.h> in your solution.

**char * myStrchr( const char *s, char ch )**

```
char *myStrchr(const char *s, char ch) {
char *cp=s;

  while (*cp!='\0') {
    if (*cp == ch) break;
    cp++;
  }
  if (*cp == '\0') cp = NULL;
  return cp;
}
```

36. [11 marks] Write a C function called **findLast** that finds the final occurrence of the string referenced by **searchStr** in the string referenced by **s**, and returns a pointer to this occurrence; if **searchStr** is not found, the function returns the null pointer instead. You ***may use*** functions from <string.h> in your solution.

**char * findLast( const char *s, const char * searchStr )**

```
char *findLast(const char *s, const char *searchStr) {

char *sp=NULL;
int ls= strlen(s), lss=strlen(searchStr);

  for (int i=ls-lss; i>=0; i--) {
    if (strncmp(s+i, searchStr, lss)==0) {
      sp = s+i;
      break;
    }
  }
  return sp;
}
```

37. [5 marks] Describe in your own words the action that is performed by the following Bourne shell script.

```
#!/bin/sh
for k in *
do
   if [ -f $k ]; then
      str="`grep  $1  $k`"
      if [ -n $str ]; then
         rm –f $k
      fi
   fi
done
```

The shell script checks regular files (not a directory) in the current directory. These  that contain string specified by the first argument will be deleted.

38. [5 marks] Describe in your own words the action that is performed by the following Bourne shell script.

```
#!/bin/sh
if test $#  –gt  0
then
   if test  –f  $1
   then
      echo $1
   fi
   shift
   $0  $*
fi
```

The shell script checks the script arguments in order.  If it is a regular file in the current directory, then print it to the standard output.

39. [12 marks] Write a Bourne shell script that counts and reports the number of regular files and the number of subdirectories found in the current working directory.

```
#!/bin/sh
countf=0
countd=0
for i in *; do
  if test –f $i; then
    countf=`expr $countf + 1`
  fi
  if test –d $i; then
    countd=`expr $countd + 1`
  fi
done
echo Total of $countf regular files.
echo Total of $countd directories.
```

40. [10 marks] Write a Bourne shell script that prints out all of its command line parameters that begin with the letter **A**.