

1.) [12 points] Short Answer Questions, Write a short paragraph – complete sentences and thoughts -- to answer each of the questions given here.

a) Describe the parts of and the purpose of the Process Control Block (PCB).

b) Describe Ahmdal's Law and how it can be used to make your code more efficient.

- c) Describe the purpose of the "Retirement Unit", which is part of the Instruction Control Unit.

- d) Show how the following code could be made more efficient for instruction level pipelining assuming you want at least 3 instructions before accessing the same register so that you can keep the pipeline filled. Also, explain why it is more efficient.

```
int    sum=0, arr[100], i;
/*arr is completely filled here */
for (i = 0; i < 100; i++) {
    sum += arr[i];
}
```

2.) [19 points] Write functions to do each of the following tasks using the following types:

<pre>#define TSIZE 27 typedef struct Node { int *key; int *value; struct Node *next; } Node; typedef Node* Pntr;</pre>	<pre>typedef Pntr[TSIZE] TArr; typedef struct{ int size; /* total number of elements in table */ TArr data; /* actual data of the table */ } Table; /* an empty table has a 0 for the size and all pointers in the array as NULL */</pre>
------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

- a) Write the function which will completely empty the hash table (leaving an empty table). You must be sure there are no memory leaks, and you do not have any helper functions provided that will clean up parts of the table. The function returns 0 if there were nodes to delete or -1 if the table is empty or any error occurred.

FUNCTION CALLED: Table x; /*other code here filling the table*/ cleantable(&x);

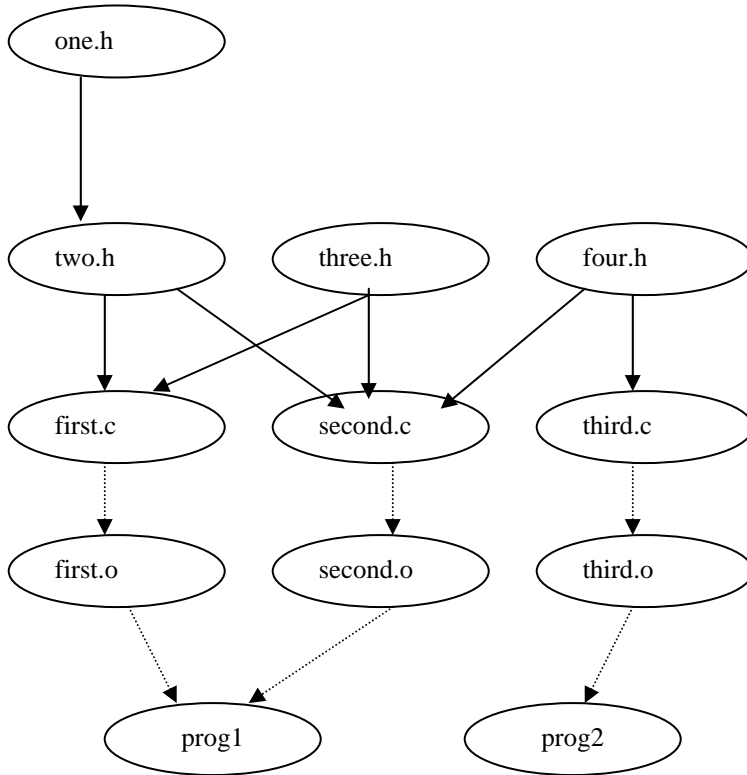
PROTOTYPE: int cleantable(Table*);

- b) Write a function that creates a new chained (bucket) hash table which is an exact independent copy of the original. The first parameter passes in the hash table to be copied while the second passes back the new copy that you must completely create. The function should not modify the original table. It should insert all elements into the same hash line on the table (it does not need to rehash), but the value field should be increased by one. Since the hash was based on the key field rather than the value, changing the value field will not change where it is hashed to. It should return as the return value of the function 0 if the table is completely empty (but any empty copy was made anyway), or the number of elements in the largest chain of the hash table if there are one or more elements in the table, or a -1 for any type of failure.

FUNCTION CALLED: Table x,*y=NULL; /*other code filling x; y is still NULL;*/ copyhash(x,y);

PROTOTYPE: int copyhash(Table, Table *);

- 3.) [12 points] Write a makefile that builds the programs with the following dependencies. A solid line indicates a file included by another file. You must write the makefile in such a way that typing the command `make` at the command line will create both executable files based on the correct versions of the files in the current directory. You must make sure you compile so that the code is optimized as much as possible, that you can use the debugger and so all warnings are noted and treated as errors.



- 4.) Assume you wish to use a binary search tree to store a list of three functions for each of several different people.
- a) [8 points] You need to define the types needed to do this. Each function returns a float, and takes as parameters an array of integers and an integer that specifies the number of used elements in that array. You only need to create the types necessary for storage as described below.
1. Create a type named `FPntr` that is the function pointer to an individual function of the type described above.

 2. Create a type named `FArr` that is the array of three function pointers.

 3. Create a type named `DataTy` that is the data (name and array of function pointers).

 4. Create a type named `Node` that is appropriate for storing the `DataTy` instances in a binary search tree.

- b) [12 points] These questions continue from the previous page. Using the types you just defined, give prototypes for the following functions:
1. NameHere() - parameters are a character array and a pointer to the root of the tree (as defined above) ; returns a 1 if the name passed as the character array appears in the tree or a 0 if it does not.
 2. AddData() - takes a DataTy as the first argument and a second parameter that allows the root pointer to be read and modified by the function. The contents of the DataTy should be added into the tree in the appropriate place for a binary search tree. The function's return type should report the value -1 if there was any type of error during the insert or 0 if everything was fine.
 3. NewData() - parameters are a character array and an array of function pointers; returns a pointer to a DataTy. The data contained in the parameters is be used to create the return value.

5.) [12 Points] For each of the following blanks, tell what the value of the variable would be immediately after the line containing the blank is executed

or if you don't know an exact numerical value, describe what value it would be in English
or write "???" for any values requested that you would not know the value of (this includes values you can not be sure of)
or write "DANGER" if referencing that variable to find out its value could cause a segmentation fault or other runtime error.

```
#include <stdio.h>
int val;

int funct1(int *q){
    q = 99;
    return q;
}
int funct2(int *a, int p){
    val = p + *a;
    p = 75;
    *a = 24;
    return val;
}
int main(int argc, char * argv[]){
    int a = 7, *p, b, c = 9, *q = &c;
    int *arr =(int*)calloc(sizeof(int),1);

    b = funct1(&a);           // b= _____

    b = c++;                  // a= _____

    *arr= &c;                 // *arr = _____
                               // arr = _____

    a = 14;
    p = &a;
    funct2(p,a);             // a = _____
                               // *p = _____

    arr = q;                 // *arr = _____
                               // arr = _____

    arr[0] = 12;
    arr++;                   // c = _____
                               // *arr = _____

    arr = a++;               // *arr= _____
                               // a = _____

    return 0;
}
```

}

6.) [10 Points] For each of the following fill in the correct word from this list (not all words in the list will be used and a word in the list can be used more than once).

cylinder	cache	epoch
pipeline stall	AVL tree	normalization
branch prediction	code motion	dynamic memory
multi-programming	loop splitting	interval time
best fit allocation	instruction level pipelining	first fit allocation
register spilling	column major order	superscalar processors
array descriptor	row major order	NONE OF THOSE LISTED

- a) _____ The processor makes an educated guess as to which instruction will be done next in order to start the processing of that selected command before it knows for sure if it will be the next command done.
- b) _____ Changing the source code so that any statements that do not need to be done on each and every iteration are no longer inside the loop and the application will take less time to run.
- c) _____ Time is measured in the number of seconds since this selected time.
- d) _____ The algorithm says to find the unit closest in size without being smaller than what is requested and then take the portion of that block as needed.
- e) _____ The instruction is broken into a series of individual tasks such as fetching the instruction, decoding the instruction and executing the instruction so that different steps can be done by different functional units at the same time.

This page intentionally blank - you may continue an answer here, but it must be clearly marked both here and where the answer is continued from.