Fundamentals of Computing and Programming

Backpaper Examination. 70 marks, Time Limit 3 hrs

December 30, 2022

1. What is printed in each case of the given code snippet: [2x5=10]

   (a)
   ```
    int n=3;
    for(int i=0;i<n;i++) {
      for(int j=0;j<i;j++)
        printf("%d ",i);
      printf("\n");
   }
   ```

   (b)
   ```
   int g=100;
   void f(){
       g=200;
   }
   void h(){
       printf("%d\n",g);
   }
   void main(){
      int g=300;
      f();
      printf("%d\n",g);
      h();
   }
   ```

   (c)
   ```
   int i=1,  j=1;
      printf("%d %d ",i++, ++j);
      printf("%d %d",i ,j);
   ```

   (d)
   ```
   float f= 6.023;
   int i = f/2;
   int j = f*2;
   printf("%d %d\n",i,j);
   ```

(e)
```
int i=5; int j=10, int t;
int *p = &i;
int *q = &j;
int *p_old=p;
int *q_old=q;
t = i;  i = j;  j = t;
if(p_old == p )
    printf("SAME");
else
    printf("DIFF");
```

2. Answer briefly each of these: [2x5=10]

   (a) A structure variable `v` has size 1000 bytes. Another variable `p` points to this structure. What is likely to be the size of the variable `p` and why?

   (b) Why is it wrong for a function to return a pointer to a local variable?

   (c) We have a requirement for a variable defined inside a function to retain its values across function calls. How do we do define such a variable. Give an example of such a definition.

   (d) Is the following statement **true** or **false** "Given two pointers p and q, `(*p==*q)` is true then `(p==q)` is true." Why?

   (e) Why is the following piece of code erroneous:

   ```
   int *p;
   *p=128;
   ```

3. [3+3+4=10]

   (a) **Declare a structure** called `struct location` with two fields: (i) `latitude` and (ii) `longitude` which are floats; **Declare another structure** called `struct address` which contains three fields (i) and integer `house_no`, (ii) a character array `street` which is a string of length at most 20 and (iii) a field called `loc` which is a `struct location` defined earlier.

   (b) Define a function `void read_address(struct address * a)` which has a parameter which points to an address structure. Into that structure it reads from the user and fills in the structure pointed to by `a` .

   (c) Write out the above `compare` function. It compares the two `struct address` pointed to by its two parameters. It returns 1 if they have identical information and returns 0 otherwise.

4. Consider the following definition: [2+5+3=10]

```
struct point {
   float x;
   float y;
};
```

(a) Declare a variable which is an array of at most 100 *pointers* to such `struct point`.

(b) Write out a function with this prototype: `void bubble_sort(struct point * a[], int n);` It takes an array of n pointers to `point` structures and sorts them in increasing order using the bubble sort algorithm. They are to be sorted by the `x` field.

(c) Using the big-O notation mention the analysis for the number of moves and number of compares in your code. Say how you arrived a your analysis.

5. Consider this declaration: [2+4+2=10]

```
struct node {
    int v;
    struct node *next;
};
```

(a) Assume we have a list of `n` integer values stored in ascending order. We wish to search for a certain value in the list. Which is better - an array of integers or a linked list using the above declaration? Explain by mentioning the time complexity of searching. What searching method would you use in each case?

(b) Write a function to delete a node from the linked list. It returns a pointer to the head of the modified linked list. It's parameters are the pointer to the head of a linked list and an integer value that needs to be searched. The corresponding node deleted if it is in the list. Otherwise the list is left unchanged. You may assume no repeated values exist in the linked list.

(c) If the linked list has values in sorted order, write an function `struct node * append_value(struct node * head, int val);` that will add a new node with value `val` at the end of the linked list, given the pointer to the head of the linked list. Note that head could be NULL if the list is empty. It returns a pointer to the head of the modified linked list.

6. (a) This question is about binary search. [2+6+2=10]
When (what kind of data and what kind of data variable) is binary search useful. What is its complexity where n is the number of data items?

(b) Write out binary search function given an array of integers and the number of integers in the array as parameters. It returns the index of the element if it is in the array. It returns -1 otherwise.

(c) An array has n elements. It is divided into k equal parts (k is some constant, independent of n). Each of the k parts is sorted. How can we use binary search to search for an element in the array? what is the total time complexity of the search?

7. Consider the following function which computes factorial for a positive number n: [2+3+5=10]

```
1 int fact(int n){
2    if (n==1) return 1;
3    int prod=0;
4    prod=fact(n-1)*n;
5    printf("%d %d\n",n,prod);
6    return prod;
7 }
```

(a) How many times is the function `fact()` invoked if `main()` called `fact(4)` and with what values and in what order?

(b) Assume that `main()` calls `fact(4)`. Show the call stack with values of `n` and `prod` for each invocation of fact on the stack when line 5 is being executed and `n` is 3.

(c) We know the classical definition of Fibonacci numbers: $F_0 = 1; F_1 = 1; F_n = F_{n-1} + F_{n-2}\ \forall(n \geq)2$. Write a recursive function to return the Fibonacci number F_n given the integer n as the parameter using the above recursive formulation.