Exam 05 solution

Question No. 01
Explain the pass by value and pass by reference mechanisms. Give examples that show their difference.

Answer:
In Pass by value mechanism, the value of a function parameter is copied to another location in the memory. When the program access the variable within the function, it can modify or change the copy only, the original variable value remains unmodified.
On the contrary, in the pass-by address mechanism, the memory address of a variable is passed to the function, and the function can access, and modify the original variable.

Code example:

```
int main()
{
   int num1 = 5, num2 = 10;
   int num3= 20;

   // address of num1 and num2 is passed
   swap( &num1, &num2);
   printf("num1 = %d\n", num1);
   printf("num2 = %d\n", num2);

   // pass by value
   square(num3);
   printf("in main function, num3 = %d\n", num3);

   return 0;
}
void square(int num3)
{
   num3=num3*num3;
   printf("inside function, num3=%d\n",num3);
}
void swap(int* n1, int* n2)
{
   int temp;
   temp = *n1;
   *n1 = *n2;
```

```
   *n2 = temp;
}
```

Output:
num1 = 10
num2 = 5
inside function, num3=400
in main function, num3 = 20

Differences:

In the example above, num1 and num2 are passed by reference mechanism in the swap function and num3 is passed by value mechanism to the square function.

1. Pass-by by value makes a copy of the variable and pass-by by reference passes the addresses of the variables to the function. Here, addresses of num1 and num2 are passed to the swap function, and the value of num3 is passed to the square function.
2. In pass by value mechanism, changes made in the function do not reflect in the original variable value, in pass by reference, changes are reflected in the original value. In the example above, the original variables num1 and num2 are modified by the swap function, but num3 is not modified by the square function.
3. In pass by value, changes made in the function remain in that scope, but this does not happen with pass by reference mechanism. In the example, we get the square value of num3(400) in the square function but in the main function, num3 still holds the original value of 20.

Question No. 02

Consider the function -

```
int f(int n, int a[]) {
    Int cnt = 0;
    for (int i=0; i<n; i++) {
       if (a[i] == a[0]) cnt++;
   }
   return cnt;
}
```

Explain what it does in one sentence. What is the return value when n = 5 and a = {1, 2, 1, 2, 1}?

Answer:

This function "f" receives an integer type array along with its size and returns the number of occurrences of the 0^{th} element of the array (here, a[0]), moreover, this function returns an integer number to the main function.

The return value is 03 because 1 occurs 3 times in the array a.

Code snippet:

```
//function
int f(int n, int a[]) {
     int cnt = 0;
     for (int i=0; i<n; i++) {
         if (a[i] == a[0]) cnt++;
    }
    return cnt;
}
// driver code
int main(){
    //int a[n];
    int n=5;
    int a[5]={1, 2, 1, 2, 1};
    int res=f(5,a);
    printf("%d",res);
   return 0;
}
```

Question No. 03
Implement the makeStrCopy function. Remember that, It takes a string in copies to an output string out. The signature should be void makeStrCopy(char in[], char out[]). For example - if in = “hello”, after calling makeStrCopy, out should also be “hello”

Answer:
 	
#include<stdio.h>
#include<string.h>
 	
//function
void makeStrCopy(char in[], char out[]) {
 	
  //copies "in" string to "out" string
  strcpy(out,in);
 	
  // //another method
  // int sz=strlen(in);
  // printf("%d",sz);
  // for(int i=0; i<sz; i++){
  // out[i]=in[i];
  // }
  // out[sz]='\0';
 	
}
// driver code
int main(){
  char in[100]="hello tesla";
  char out[100];
 	
  makeStrCopy(in,out);
  printf("out= %s",out);
 	
  return 0;
}

Question No. 04
Dynamically allocate an array of floats with 100 elements. How much memory does it Take?
Answer: Here, a single float type variable requires 4 bytes of memory, then the array will reserve 4*100=400 bytes of memory. Declaration: float *a=(float*)malloc(100*sizeof(float)); Sizeof(float) is 4 bytes, and the pointer holds the address of the first byte in the allocated memory. Heap is used for dynamic memory allocation.

Question No. 05
Suppose int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9}. Suppose the address of a[0] is at 6000. Find the value of the following - a. a[8] b. &a[5] c. a d. a+4 e. *(a+2) f. &*(a+4)

Answer:
a. 9
b. 6020
c. 6000
d. 6016
e. 3
f. 6016
 	
<u>Related code:</u>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
 	
// driver code
int main(){
  int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
 	
  for(int i=0; i<9; i++){
  printf("%lld %d\n",(a+i),*(a+i));
  }
 	
  printf("1. %lld\n",a[8]);// value of a[8]
  printf("2. %lld\n",&a[5]); //add of a[5]
  printf("3. %lld\n",a); //add of a[0]
  printf("4. %lld\n",a+4);// add of a[4]
  printf("5. %lld\n",*(a+2)); //value of a[2]
  printf("6. %lld",&*(a+4)); // add of a[4]
 	
  return 0;
}

Question No. 06

Ash tries to implement bubble sort the following way. In particular, notice that the loop iterates on the array in reverse. Fill in the box to implement the function.

```
void sort(int n, int a[]) {
    for (int steps=0; steps<n; steps++) {
        for (int i=n-1; i>0; i--) {
            ///Write code here
        }
    }
}
```

Answer:

Code:

```
#include<stdio.h>

//function
void  swap(int *a, int *b){
    int temp=*a;
        *a=*b;
        *b=temp;
}
void  sort(int n,int a[]){
    for(int step=0; step<n; step++){
        for(int i=n-1; i>0; i--){
            if(a[i]<a[i-1]){
                swap(&a[i] ,&a[i-1]);
            }
        }
        printf("steps:%d\n",step);
        for(int i=0; i<n; i++){
            printf("%d ",a[i]);
        }
        printf("\n");
    }
}
// driver code
int main(){
    int sz;
    scanf("%d",&sz);
    int array[sz];

    for(int i=0; i<sz; i++){
```

```
        scanf("%d",&array[i]);
    }
    sort(sz,array);
    printf("after sorting:");
    for(int i=0; i<sz; i++){
        printf("%d ",array[i]);
    }
   return 0;
}
```

Question No. 07
Implement the is_reverese_sorted() function to check if an array reverse sorted. For example if a = {6, 4, 3, 1}. Then is_reverse_sorted should return True.

Answer:

Code:

```
#include<stdio.h>
#include<stdbool.h>

bool is_reverese_sorted(int sz,int a[]){
    int flag=0;
    for(int i=0; i<sz-1; i++){

        if(a[i]<=a[i+1]){
            flag=1;//not reverse sort
        }
        //printf("%d %d %d %d %d\n",i,i+1,a[i],a[i+1],flag);
    }
    if(flag!=0) return false;
    else return true;
}

// driver code
int main(){
    int a[4]={6, 4, 3, 1};
    int sz=sizeof(a)/sizeof(a[0]);

    int res=is_reverese_sorted(sz,a);
    if(res==true) printf("reverse sorted");
    else if(res==false) printf("not reverse sorted");
    return 0;
}
```

Question No. 08
Modify the Selection sort function so that it sorts the array in reverse sorted order, ie. from the largest to smallest. For example reverse sorting a = {3, 4, 2, 5, 1} should result in {5, 4, 3, 2, 1}. Use the is_reverse_sorted() function to break early from the function if the array is already sorted

Answer:
Full code:

```
#include<stdio.h>
#include<stdbool.h>

void  RevSelectionSort(int array[],int sz);
void  swap(int *a, int *b);
bool is_reverese_sorted(int sz,int a[]);

// driver code
int main(){
    int sz;
    scanf("%d",&sz);
    int array[sz];

    for(int i=0; i<sz; i++){
        scanf("%d",&array[i]);
    }
    // for(int i=0; i<n; i++){
    //     printf("%d ",array[i]);
    // }
    RevSelectionSort(array,sz);
    printf("\nsorted array:");

    for(int i=0; i<sz; i++){
        printf("%d ",array[i]);
    }
    return 0;
}
//function
void  RevSelectionSort(int array[],int sz){
    for(int step=0; step<sz; step++){
        int max=array[step], pos=step;
```

```
        for(int i=step; i<sz; i++){
            if(array[i]>max){
                max=array[i];
                pos=i;
            }
        }
    swap(&array[step],&array[pos]);
    // sort checking
    printf("\nsteps: %d\n ",step);
    printf("max value: %d found max value: %d\n",array[pos],array[step]);
    for(int i=0; i<sz; i++){
        printf("%d ",array[i]);
    }
    //is already sorted??
    int res=is_reverese_sorted(sz,array);
    if(res==true){
        printf("sorted done\n");
        break;
    }
    else if(res==false) printf("not sorted\n");
    }
}
void  swap(int *a, int *b){
    int temp=*a;
        *a=*b;
        *b=temp;
}
bool is_reverese_sorted(int sz,int a[]){
    int flag=0;
    for(int i=0; i<sz-1; i++){

        if(a[i]<=a[i+1]){
            flag=1;
        }
        //printf("%d %d %d %d %d\n",i,i+1,a[i],a[i+1],flag);
    }
    if(flag!=0) return false;
    else return true;
}
```

Question No. 09

We wrote a program to find all positions of a character in a string with the strchr function. Now do the same without using strchr.

Answer:

Code:

```
#include<stdio.h>

// driver code
int main(){
    char str[]="proprogrammipngp";
    char ch='p';

    int len=strlen(str);

    for(int i=0; i<len; i++){
        if(str[i]==ch){
            printf("p is at index %d\n",i);
        }
    }
    return 0;
}
```

Output:

```
p is at index 0
p is at index 3
p is at index 12
p is at index 15
```

Question No. 10
Is there any difference in output if you call strstr(text, “a”) and strchr(text,‘a’)? Explain with examples.

Answer:
Here, strstr is used to find a substring in a string, and strchr is used to find a character in a string.
strstr returns the first occurring address of the substring “a” from the string “text”.
strchr returns the first occurring address of the character ‘a’ from the string “text”.
 	
The process is explained with a coded example below,
<u>Code snippet:</u>
 	
int main(){
  char str[]="helloworld";
 	
  int len=strlen(str);
  for(int i=0; i<len; i++){
  printf("%d %c %lld\n",i, str[i],&str[i]);
  }
  char *pos=strchr(str,'l');
  printf("%s %lld\n",pos,&*pos);
 	
  char *pos1=strstr(str,"l");
  printf("%s %lld\n",pos1,&*pos1);
 	
  char *pos3=strstr(str,"lo");
  printf("%s %lld\n",pos3,&*pos3);
 	
  return 0;
}
 	
<u>Output:</u>
0 h 339663124005
1 e 339663124006
2 l 339663124007
3 l 339663124008
4 o 339663124009
5 w 339663124010
6 o 339663124011
7 r 339663124012

8 l 339663124013
9 d 339663124014
lloworld 339663124007
lloworld 339663124007
loworld 339663124008

explanation:
strchr returns the address of the first 'l' character (as can be seen from the address of all the characters.) and then prints all the subsequent characters (lloworld).
strstr considers "l" as a substring and returns the address of the first 'l' character then prints all the subsequent characters (lloworld).
In the third case, strstr considers "lo" as a substring and returns the address of the second 'l' character because "lo" is found in string index 3 and 4. That's why this time strstr starts printing from index 3 and prints "loworld".

---------- **END** ----------

Date:27th May, 2022

Submitted by: subrata saha

Email: subratabaec@gmail.com