



Level: 1st year computer science
Material: ADS2

TD/TP Typical Answer N^o: 03

Academic year: 2023/2024
Chapter 1 : Pointers

Exercise 1: (TD)

Complete the following table that shows the value of each variable after each statement.

| instruction | a | b | c | p1 | p2 |
|------------------------|----|---|---|----|----|
| int a, b, c, *p1, *p2; | / | / | / | / | / |
| a=1; b=2; c=3; | 1 | 2 | 3 | / | / |
| p1=&a; p2=&c; | 1 | 2 | 3 | &a | &c |
| *p1=(*p2)++; | 3 | 2 | 4 | &a | &c |
| p1=p2; p2=&b; | 3 | 2 | 4 | &c | &b |
| *p1--=*p2; | 3 | 2 | 2 | &c | &b |
| instruction | a | b | c | p1 | p2 |
| ++*p2; | 3 | 3 | 2 | &c | &b |
| *p1**=*p2; | 3 | 3 | 6 | &c | &b |
| a++*p2**p1; | 24 | 4 | 6 | &c | &b |
| p1=&a; | 24 | 4 | 6 | &a | &b |
| *p2=*p1/=*p2; | 6 | 6 | 6 | &a | &b |

Exercise 2: (TP)

Write a program that fills an array T with real numbers, then creates two arrays TP and TN, and puts all the positive numbers in TP and all the negative numbers in TN, and leaves the null numbers as they are.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int *T,*TP,*TN, n, i, j, k;
    printf("enter nbr of elements\n");
    scanf("%d",&n);
    T=(int*)malloc(n*sizeof(int));
    for( i=0; i<n; i++) {
        printf("T[%d]=",i);
        scanf("%d",T+i);
    }
    TP=(int*)malloc(n*sizeof(int));
    TN=(int*)malloc(n*sizeof(int));
    j=k=0;
    for( i=0; i<n; i++)
        if(T[i]>0)
            TP[j++]=T[i];
        else if(T[i]<0)
            TN[k++]=T[i];
    TP=(int*)realloc(TP, j*sizeof(int));
    TN=(int*)realloc(TN, k*sizeof(int));
    printf("\nthe table of positives\n");
    for( i=0; i<j; i++)
        printf("%d\t",TP[i]);
    printf("\nthe table of negatives\n");
    for( i=0; i<k; i++)
        printf("%d\t",TN[i]);
    return 0;
}
```

Exercise 3: (TD) Let p be a pointer pointing to array T:

```
int T[] = {8, 17, 7, 9, 48, 76, 22, 27};
int *p =T;
```

What is the value or address returned by each of the following expressions:

| | |
|---------------------|--------------------------|
| 1. *p+3 | T[0]+3 ⇒11 |
| 2. p+(*p-7) | p+(T[0]-7) ⇒p+1⇒&T[1] |
| 3. &T[5]-p | p+5-p⇒5 |
| 4. &p+1 | &p+1 |
| 5. *(p+3) | *(&T[3]) ⇒9 |
| 6. *(P+*(P+7)-T[6]) | *(p+T[7]-T[6]) ⇒T[5] ⇒76 |
| 7. T+2 | p+2⇒&T[2] |
| 8. &T[5]-2 | p+5-2⇒p+3⇒&T[3] |

Exercise 4: (TD/TP)

Write the *strcat* function, which concatenates two strings into a new string. (use the * dereferencing operator instead of []).

```
#include <stdio.h>
#include <stdlib.h>
char * strcat(char *s1, char *s2) {
    char *s;
    int i=0, n=strlen(s1)+strlen(s2)+1;
    s=(char*)malloc(n*sizeof(char));
    while(*s1){
        *(s+i)=*s1;
        i++; s1++;
    }
    while(*s2){
        *(s+i)=*s2;
        i++; s2++;
    }
    *(s+i)='\0';
    return s;
}
int main() {
    char *s;
    char str1[100] = "Hello,";
    char str2[100] = "world!";
    s = strcat(str1, str2);
    printf("After concatenation: %s", s);
    return 0;
}
```

Exercise 5: (TP)

Write the *copy* function that copies part of an array. The function takes the array and its copied size, start, and length and returns a pointer to the new array.



```
#include <stdio.h>
#include <stdlib.h>
int * copy(int *t,int n,int pos,int size)
{ int i, *c;
  If (pos+size>n) return NULL;
  c=(int*)malloc(size*sizeof(int));
  for(i=0; i<size; i++)
    c[i]=t[i+pos];
  return c;
}
int main() {
  int i, pos=2, size=3;
  int t[] = {1, 2, 3, 4, 5, 6};
  int *c = copy(t, 6, pos, size);
  if(c !=NULL){
    for (i = 0; i < size; i++)
      printf("%d ", c[i]);
    free(c);
  }
  return 0;
}
```

Exercise 6 : (TD/TP)

Suppose we have a black and white image of dimensions $n \times m$, stored as a matrix of integers.
 Write a function called "*flip*" that flips the image vertically

```
#include <stdio.h>
#include <stdlib.h>
int ** flip(int **A,int n,int m) {
  int i, j, **T;
  T=(int**) malloc( n*sizeof(int*));
  for(i=0; i<n; i++)
    T[i]=(int*) malloc(m*sizeof(int));
  for(i=0; i<n; i++)
    for(j=0; j<m; j++)
      T[i][j]=A[n-i-1][j];
  return T;
}
```

```
void flip2(int **B, int n) {
  int i, *lin;
  for(i=0; i<=n%2; i++) {
    lin=B[i];
    B[i]=B[n-i-1];
    B[n-i-1]=lin;
  }
}
```

```
int main() {
  int i, j, n = 2, m = 3,**A,**B;
  create A
  A = (int**)malloc(n*sizeof(int*));
  for (i = 0; i < n; i++)
    A[i] = (int*)malloc(m*sizeof(int));
  //Fill In A
  for ( i = 0; i < n; i++)
    for ( j = 0; j < m; j++)
      A[i][j] = i*m + j + 1;
  //show A
  printf("Before flip:\n");
  for ( i= 0; i < n; i++) {
    for ( j = 0; j < m; j++)
      printf("%d ", A[i][j]);
    printf("\n");
  }
  B= flip(A, n, m);
  printf("After flip:\n");
  for ( i= 0; i < n; i++) {
    for ( j = 0; j < m; j++)
      printf("%d ", B[i][j]);
    printf("\n");
  }
  //delete A
  for (i = 0; i < n; i++)
    free(A[i]);
  free(A);
  //delete B
  for (i = 0; i < n; i++)
    free(B[i]);
  free(B);
  return 0;
}
```

Exercise 7: (TD)

Write the *Mat2Tab* function that converts a matrix into an array.



```
#include <stdio.h>
#include <stdlib.h>
int* Mat2Tab(int**M,int n, int m) {
    int i, j, k=0, *T;
    T=(int*)malloc(n*m*sizeof(int));
    for( i=0; i<n; i++)
        for( j=0; j<m; j++) {
            T[k]=M[i][j];
            k++;
        }
    return T;
}
int main() {
    int i, j, n = 2, m = 3,**mat,*T;
    //Create Matte
    mat = (int**)malloc(n*sizeof(int*));
    for (i = 0; i < n; i++)
        mat[i] = (int*)malloc(m*sizeof(int));
    //Fill in mat
    for ( i = 0; i < n; i++)
        for ( j = 0; j < m; j++)
            mat[i][j] = i*m + j + 1;
    //Show matte
    printf("2D array:\n");
    for ( i= 0; i < n; i++) {
        for ( j = 0; j < m; j++)
            printf("%d ", mat[i][j]);
        printf("\n");
    }
    //Transform Matte to 1D
    T = Mat2Tab(mat, n, m);
    //show T
    printf("1D array:\n");
    for ( i = 0; i < n*m; i++)
        printf("%d ", T[i]);
    //delete T
    free(T);
    //Delete Matte
    for (i = 0; i < n; i++)
        free(mat[i]);
    free(mat);
    return 0;
}
```