SM2-21st C Programming

Tutorial 3

27 Oct 2017, Friday 8am, LT24

You are to provide the solution in LT.

1. What are the considerations given to the analysis of the time performance of algorithms?

2. Give *n* data, what is the worst runtime of Heap Sort? Explain and derive your answers.

3. The contents of a linear array a[8] is as follows:

   ```
   5  3  8  9  1  0  2  6
   ```

   where a[0] = 5, a[1] = 3, …, and a[7]=6. Assume that the array is to be sorted in non-decreasing order, i.e., a[0] ≤ a[1] ≤ a[2] ≤...≤ a[7]. Write down the contents of the array after the first iteration of Bubble Sort algorithm has been executed.

4. The author of the following program claims that the following program contains the algorithm of Heap Sort.

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

void main()
{
  int *x,i,n;
  int temp;
  void heap (int *,int);
  long int checksum;

  printf("\n\nHeap Sort:\n\n");
  printf("How many Numbers : ");

  scanf("%d",&n);
  x = (int *) malloc (n * sizeof(int));

  srand(13);  // fix the seed
  for(i=0;i<n;i++) x[i]= rand()%1000;

  checksum=0;
  for(i=0;i<n;i++) checksum += x[i];
  printf("\nBefore:");
  for(i=0;i<n;i++) printf(" %d ",x[i]);
  printf("\nchecksum = %ld.",checksum);

  heap(x,n);


  for(i=n-1;i>=1;i--)
  {
    temp =  x[i];
    x[i] = x[0];
    x[0] = temp;
    heap(x,i-1);
  }
```



```
  printf("\nAfter:");
  for(i=0;i<n;i++) printf(" %d ",x[i]);

  checksum=0;
  for(i=0;i<n;i++) checksum += x[i];
  printf("\nchecksum = %ld.",checksum);

   getch();
}

void heap(int *a,int n)
{
  int i,temp;

  for(i=n/2;i>=0;i--)
  {
    if(a[(2*i)+1] < a[(2*i)+2] && (2*i+1)<n && (2*i+2)<n)
    {
      temp = a[(2*i)+1];
      a[(2*i)+1] = a[(2*i)+2];
      a[(2*i)+2] = temp;
    }

    if(a[(2*i)+1] > a[i] && (2*i+1)<n && i<n)
    {
      temp = a[(2*i)+1];
      a[(2*i)+1] = a[i];
      a[i] = temp;
    }
  }
}
```

(i) Explain why you **disagree** that the algorithm used in the program is Heap Sort.

(ii) Given *n* data to be sorted, what is the order of the runtime of the algorithm used in the program. Explain how you derive the answer.

5. Given the following declaration

```
char S[] = "abcdefg";
```

What is output by the instruction

```
printf ("%s", S+3); ?
```

6. What is the output of the following program ?

```
# include <stdio.h>

/* passing by value and reference */

void change (int a, int *b)
{
  *b = a;
  a=1;
  b=&a;
  printf ("\n In function change : a=%d, *b=%d", a,*b);
}

main()
{
```

```
  int s=2, t=3;

  printf ("\n In function main : s=%d, t=%d", s, t);
  change (s, &t);
  printf ("\n In function main : s=%d, t=%d", s, t);

  return 0;
}
```

7. Given the following declaration

```
int A[] = {5, 4, 3, 2, 1};
```

What is the value of the expression **`* A+1`** ?

8. What is the output of the following code segment ?

```
int A[5] = {3, 0, 4, 2, 1}, i;
for (i=0; i<5; i++)
     printf ("%d ", A[A[i]]);
```

9. Given the following declaration and initialisation:

```
int a=7;
```

Do the following code segments produce the same result? Why?

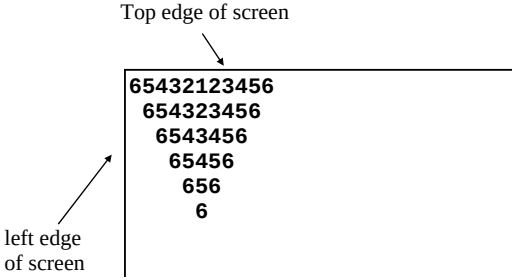
```
if (a==2)
  printf (" a=2 ");
else
  printf (" a!=2 ");
```

Segment 1

```
if (a==2) printf (" a=2 ");
if (a<2)  printf (" a!=2 ");
```

Segment 2

10. Write a C program to produce the following number pattern on the screen by a **nested loop**.



11. What is the screen output of the following program ?

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



```
int main()
{
  char *a, *b, *c, *d, w='w',x='x',y='y',z='z';

  a = &w;
  b = &x;
  c = &y;
  d = &z;

  a=b;
  b=a;
  c=b;
  d=c;

  printf ("%c %c %c %c",*a,*b,*c,*d);

  return 0;
}
```

12. Given the following declarations :

```
int i, j, k;

int A[4] [5] = { 1, 2, 3, 4, 5,
                 2, 3, 4, 5, 6,
                 3, 4, 5, 6, 7,
                 4, 5, 6, 7, 8  };

int B[5] [2] = {  1, 2,
                  3, 4,
                  5, 6,
                  7, 8,
                  9, 0  };

int C[4] [2];
```

where **`A`**, **`B`** represent the contents of two matrices of sizes 4x5 and 5x2 respectively.

Without declaring any additional variable, write down the necessary C statements to multiply matrices **`A`** and **`B`** and store the product in the array **`C`**.