

CP 111 Computer Programming

2/2554 Section B01/B02

Lab 9 Solution

1. Problem name: Pos_Neg_Avg_Std_Func_Pointer_Offset_Notation

```
#include <stdio.h>
#include <math.h>
void Pos_Neg_Avg_Std_Func(int *X, int N, double *pos_avg, double *pos_std,
                        double *neg_avg, double *neg_std);

int main()
{
    int N,x,data[100];
    int *ptr;
    double pos_avg,pos_std,neg_avg,neg_std;
    N=0;
    ptr=data;
    while(1)
    {
        scanf("%d",&x);
        if(x==0)
            break;
        *ptr=x;
        ptr++;
        N++;
    };
    Pos_Neg_Avg_Std_Func(data,N,&pos_avg,&pos_std,&neg_avg,&neg_std );
    printf("%.2f %.2f\n",pos_avg,pos_std);
    printf("%.2f %.2f\n",neg_avg,neg_std);
    return 0;
}

void Pos_Neg_Avg_Std_Func(int *X, int N, double *pos_avg, double *pos_std,
                        double *neg_avg, double *neg_std)
{
    int pos_sum,neg_sum,n_pos,n_neg,i;
    double pos_sum_sq,neg_sum_sq;
    int *ptr;
    pos_sum=0;
    neg_sum=0;
    n_pos=0;
    n_neg=0;
    ptr=X;
    for(i=0;i<N;i++)
    {
        if(*(ptr+i)>0)
        {
            pos_sum=pos_sum+*(ptr+i);
            n_pos++;
        }
        else
        {
            neg_sum=neg_sum+*(ptr+i);
            n_neg++;
        }
    }
}
```

```

    *pos_avg=(double)pos_sum/n_pos;
    *neg_avg=(double)neg_sum/n_neg;
    // Calculate Standard deviations
    pos_sum_sq=0;
    neg_sum_sq=0;
    ptr=X;
    for(i=0;i<N;i++)
    {
        if(*ptr>0)
        {
            pos_sum_sq = pos_sum_sq + (*ptr-*pos_avg)*(*ptr-*pos_avg);

        }
        else
        {
            neg_sum_sq = neg_sum_sq + (*ptr-*neg_avg)*(*ptr-*neg_avg);
        }
        ptr=ptr+1;
    }
    *pos_std=sqrt(pos_sum_sq/n_pos);
    *neg_std=sqrt(neg_sum_sq/n_neg);
}

```

2. Problem name: Sort_N

```
#include <stdio.h>
void InsertionSort(int *A,int n);
int main()
{
    int data[100];
    int N,i;
    scanf("%d",&N);
    for(i=0;i<N;i++)
        scanf("%d",&data[i]);
    InsertionSort(data,N);

    for(i=0;i<N;i++)
        printf("%d\n",data[i]);
    return 0;
}

void InsertionSort(int *A,int n)
{
    int i,j,key;
    for( j=1 ; j<n ; j++ )
    {
        key=A[j];
        i=j-1;
        while(1)
        {
            if(A[i] <= key || i<0)
                break;
            A[i+1]=A[i];
            i=i-1;
        }
        A[i+1]=key;
    }
}
```

3. Problem name: *Median*

```
#include <stdio.h>
void InsertionSort(int *A, int n);

int main()
{
    int x,N,med;
    int X[100];

    N=0;
    while(1)
    {
        scanf("%d",&x);

        if(x<0)
            break;

        X[N]=x;
        N=N+1;
    }

    InsertionSort (X,N);

    if(N%2==1)
    {
        med=X[N/2];
    }
    else
    {
        med=(X[N/2-1]+X[N/2])/2;
    }
    printf("%d\n",med);

    return 0;
}

void InsertionSort(int *A,int n)
{
    // similar to Problem Sort_N
}
```

4. Problem name: Matrix_Multiplication

```
#include <stdio.h>
#define MAX_N 20

int main()
{
    int A[MAX_N][MAX_N];
    int B[MAX_N][MAX_N];
    int C[MAX_N][MAX_N];
    int N,i,j,k;

    scanf("%d",&N);
    for(i=0;i<N;i++)
        for(j=0;j<N;j++)
            scanf("%d",&A[i][j]);

    for(i=0;i<N;i++)
        for(j=0;j<N;j++)
            scanf("%d",&B[i][j]);

    for(i=0;i<N;i++)
    {
        for(j=0;j<N;j++)
        {
            C[i][j]=0;
            for(k=0;k<N;k++)
                C[i][j]=C[i][j]+(A[i][k]*B[k][j]);
        }
    }
    for(i=0;i<N;i++)
    {
        for(j=0;j<N;j++)
            printf("%d ",C[i][j]);
        printf("\n");
    }

    return 0;
}
```

5. Problem name: Extract_word

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

```
int main()
{
    char instr[501];
    int i, state;
```

```
    gets(instr);
```

```
    state=0;
```

```
    i=0;
```

```
    while(1)
```

```
    {
```

```
        if(state==0)
```

```
        {
```

```
            if(instr[i]!=' ')
```

```
            {
```

```
                first=i;
```

```
                state=1;
```

```
            }
```

```
        }
```

```
        else if(state==1)
```

```
        {
```

```
            if(instr[i]==' ' || instr[i]==0)
```

```
            {
```

```
                last=i-1;
```

```
                state=0;
```

```
                for(k=first;k<=last;k++)
```

```
                {
```

```
                    printf("%c",instr[k]);
```

```
                }
```

```
                printf(" %d\n",last-first+1);
```

```
            }
```

```
            if(instr[i]==0)
```

```
                break;
```

```
        }
```

```
        i=i+1;
```

```
    }
```

```
    return 0;
```

```
}
```

