DS Lab 4 Solutions

4.1.1

```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<ctype.h>
#include<stdbool.h>
#define N 20
char s[20];
int top = -1;
int isEmpty()
{
    if(top<0)
    return 1;
    else
    return 0;
}
void push(char x)
{
    if(top==N-1)
    {
        printf("Stack is overflow.\n");
    }
    else
    {
        top = top+1;
        s[top]=x;
    }
}
```

```
28      char pop()
29    v {
30      ⟶if(top<0)
31    v ⟶{
32      ⟶⟶printf("Stack is underflow : unbalanced parenthesis\n");
33      ⟶⟶exit(0);
34      ⟶}
35      ⟶else
36      ⟶return s[top--];
37      }
38      int priority(char x)
39    v {
40      ⟶if(x=='(')
41      ⟶return 0;
42      ⟶if(x=='+'||x=='-')
43      ⟶return 1;
44      ⟶if(x=='*'||x=='/'||x=='%')
45      ⟶return 2;
46      }
47      bool isalphanum(char a)
48    v {
49      ⟶if((a>=65&&a<=90)||(a>=97&&a<=122))
50      ⟶return true;
51      ⟶return false;
52      }
```

```
void convertInfix(char *e)
{
    int x;
    int k = 0;
    char *p = (char*)malloc(sizeof(char)*strlen(e));
    while(*e!='\0')
    {
        if(isalphanum(*e))
        {
            p[k++] = *e;

        }
        else if(*e=='(')

        push(*e);
        else if(*e==')')
        {
            while(!isEmpty()&&(x=pop())!='(')
            p[k++]=x;
        }

        else if(*e=='+'||*e=='-'||*e=='*'||*e=='/'||*e=='%')
        {
            while(priority(s[top])>=priority(*e))
```

```
            p[k++] = pop();
            push(*e);
        }
        else
        {
            printf("Invalid symbols in infix expression. Only alphanumeric characters , parenthesis and valid arithmetic operators are allowed.\n");
            exit(0);
        }
        e++;

    }
    while(top!=-1)
    {
    x=pop();
    if(x=='(')
    {
        printf("Invalid infix expression : unbalanced parenthesis.\n");
        exit(0);
    }
    p[k++]=x;
    }
    p[k++]='\0';
    printf("Postfix expression : %s\n",p);

    }
```

4.1.2

```c
#include<stdio.h>
int main(){
    int i=0,a[100],choice,x=0;
    printf("Enter the choice for stack operation: \n");
    printf("press 1 for push\npress 2 for pop\npress 3 for display\npress 4 for exit\n");
    scanf("%d",&choice);
    while(choice!=4){
        switch(choice){
            case 1:
                printf("Enter item to insert: ");
                scanf("%d",&a[i]);
                printf("Item inserted = %d\n",a[i]);
                i++;
                break;
            case 2:
                if(i==0){
                    printf("Stack underflow\n");
                }
                else{
                    printf("Item deleted = %d\n",a[i-1]);
                    a[i-1]=0;
                    i--;
                }
                break;
            case 3:
                if(i==0){
                    printf("No more data\n");
```

```
27                printf("No more data\n");
28                break;
29            }
30            printf("Stack are\n");
31            x=i;
32            while(i>0){
33                printf("%d ",a[i-1]);
34                i--;
35            }
36            printf("\n");
37            i=x;
38            break;
39        }
40        printf("Enter the choice for stack operation: ");
41        scanf("%d",&choice);
42    }
43    }
44
45
```

4.1.3

```c
#include<stdio.h>

int read (int *a,int m,int n){
    printf("Enter %d elements : ",m*n);
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            scanf("%d",((a+i)+j));
        }
    }
}
int display(int *a,int m,int n){
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            printf("%d ",(*(a+i)+j));
        }
        printf("\n");
    }
}

int multiplicationOfTwoMatrices(int *a,int *b,int m,int n,int q){
    printf("The Multiplication Matrix is\n");
    for(int i=0;i<m;i++){
        for(int j=0;j<q;j++){
            int sum=0;
            for(int k=0;k<n;k++){
                sum+=((*(a+i)+k))*((*(b+k)+j));
            }
            printf("%d ",sum);
        }
        printf("\n");
    }
}
```

4.2.1

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct Stack{
    int top;
    unsigned capacity;
    char* array;
}Stack;

Stack* createStack(unsigned capacity){
    Stack* stack=(Stack*)malloc(sizeof(Stack));
    stack->capacity=capacity;
    stack->top=-1;
    stack->array=(char*)malloc(stack->capacity*sizeof(char));
    return stack;
}
int isFull(Stack* stack){
    return stack->top==stack->capacity-1;
}
int isEmpty(Stack* stack){
    return stack->top==-1;
}
void push(Stack* stack,char item){
    if(isFull(stack))
        return;
    stack->array[++stack->top]=item;
}
```

```
char pop(Stack* stack){
    if(isEmpty(stack))
        return -1;
    return stack->array[stack->top--];
}
int isMatchingPair(char character1,char character2){
    if(character1=='('&&character2==')'){
        return -1;

    }
    else
        return 0;
}

int areParenthesisBalanced(char exp[]){
    int i=0;
    Stack* stack=createStack(strlen(exp));
    while(exp[i]){
        if(exp[i]=='(')
            push(stack,exp[i]);
        else if(exp[i]==')'){
            if(isEmpty(stack))
                return 0;
            else if(!isMatchingPair(pop(stack),exp[i])){
                return 0;
            }
        }
```

```
        i++;
    }
    if(isEmpty(stack))
        return 1;
    else
        return 0;
}
int main(){
    char exp[100];
    printf("Enter a bracket sequence: ");
    scanf("%s",exp);
    if(areParenthesisBalanced(exp))
        printf("Balanced brackets\n");
    else
        printf("Not balanced brackets\n");
}
```

4.2.2

```c
#include<stdio.h>
#define SIZE 100


void printNGE(int arr[],int n){
    int next,i,j;
    printf("Input: ");
    for(i=0;i<n;i++){
        next=-1;
        for(j=i+1;j<n;j++){
            if(arr[i]<arr[j]){
                next=arr[j];
                break;
            }
        }
        printf("%d ",arr[i]);

    }
    printf("\n");
    printf("Output: ");
    for(i=0;i<n;i++){
        next=-1;
        for(j=i+1;j<n;j++){
            if(arr[i]<arr[j]){
                next=arr[j];
                break;
            }
        }
        printf("%d ",next);
    }
    printf("\n");
}
void input(){
    printf("Input: ");
}
int main(){
    int arr[SIZE],n,i;
    printf("Enter n: ");
    scanf("%d",&n);
    printf("Enter n elements: ");
    for(i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    printNGE(arr,n);
}
```

4.2.3

```c
#include<stdio.h>
#include<stdlib.h>
void findNextGreaterElement(int arr[],int n){
    int next[n],i,j;
    for(i=0;i<n;i++){
        next[i]=-1;
    }
    for(i=0;i<n;i++){
        for(j=i+1;j<n+i;j++){
            if(arr[j%n]>arr[i]){
                next[i]=arr[j%n];
                break;
            }
        }
    }
    for(i=0;i<n;i++){
        printf("%d ",next[i]);
    }
}
int main(){
    int n,i;
    printf("Enter n: ");
    scanf("%d",&n);
    int arr[n];
    printf("Enter n elements: ");
    for(i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    printf("Input: ");
    for(i=0;i<n;i++){
        printf("%d ",arr[i]);
    }
    printf("\nOutput: ");
    findNextGreaterElement(arr,n);
    printf("\n");
}
```