LAB-7

Implementation of Stack and Queue Data Structures and Their Applications in C

Q1) Implement a stack using an array and perform push, pop, display, and check stack overflow/underflow conditions.

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

#define MAX 5
int stack[MAX];
int top=-1;

void push(int x){
   if(top==MAX-1){
      printf("Stack Overflow\n");
      return;
   }
   stack[++top]=x;
}

void pop(){
   if(top==-1){
      printf("Stack Underflow\n");
      return;
   }
   printf("Popped element: %d\n",stack[top--]);
}

void display(){
   if(top==-1){
      printf("Stack is empty\n");
      return;
   }
   printf("Stack elements: ");
   for(int i=top;i>=0;i--)
      printf("%d ",stack[i]);
   printf("\n");
}

int main(){
   int ch,val;
   while(1){
      printf("\n1.Push  2.Pop  3.Display  4.Exit\nEnter choice: ");
      scanf("%d",&ch);
      switch(ch){
         case 1:
            printf("Enter value: ");
            scanf("%d",&val);
            push(val);
            break;
         case 2:
            pop();
            break;
         case 3:
            display();
            break;
         case 4:
            exit(0);
         default:
            printf("Invalid choice\n");
      }
   }
   return 0;
}
```

Sample Input & Output

```
1.Push  2.Pop  3.Display  4.Exit

Enter choice: 1

Enter value: 34


1.Push  2.Pop  3.Display  4.Exit

Enter choice: 1

Enter value: 34  20


1.Push  2.Pop  3.Display  4.Exit
```

```
Enter choice: 1
Enter value: 3420

1.Push  2.Pop  3.Display  4.Exit
Enter choice: 2
Popped element: 3420

1.Push  2.Pop  3.Display  4.Exit
Enter choice: 2
Popped element: 20

1.Push  2.Pop  3.Display  4.Exit
Enter choice: 1
Enter value: 2045  34

1.Push  2.Pop  3.Display  4.Exit
Enter choice: 3
Stack elements: 2034 34

1.Push  2.Pop  3.Display  4.Exit
Enter choice: 4
```

Q2) Implement a stack using a linked list and perform push, pop, peek, and display operations.

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

struct node{
   int data;
   struct node *next;
};
```

```c
struct node *top=NULL;

void push(int x){
   struct node *n=malloc(sizeof(struct node));
   n->data=x;
   n->next=top;
   top=n;
}

void pop(){
   if(top==NULL){
      printf("Stack Underflow\n");
      return;
   }
   printf("Popped element: %d\n",top->data);
   struct node *t=top;
   top=top->next;
   free(t);
}

void peek(){
   if(top==NULL){
      printf("Stack empty\n");
      return;
   }
   printf("Top element: %d\n",top->data);
}

void display(){
   if(top==NULL){
      printf("Stack empty\n");
      return;
   }
   struct node *t=top;
   printf("Stack elements: ");
   while(t){
      printf("%d ",t->data);
      t=t->next;
   }
   printf("\n");
}
```

```
int main(){
   int ch,val;
   while(1){
      printf("\n1.Push  2.Pop  3.Peek  4.Display  5.Exit\nEnter choice: ");
      scanf("%d",&ch);
      switch(ch){
         case 1:
            printf("Enter value: ");
            scanf("%d",&val);
            push(val);
            break;
         case 2:
            pop();
            break;
         case 3:
            peek();
            break;
         case 4:
            display();
            break;
         case 5:
            exit(0);
         default:
            printf("Invalid choice\n");
      }
   }
   return 0;
}
```

Sample Input & Output

```
1.Push  2.Pop  3.Peek  4.Display  5.Exit

Enter choice: 1

Enter value: 34


1.Push  2.Pop  3.Peek  4.Display  5.Exit

Enter choice: 2 1

Enter value: 20


1.Push  2.Pop  3.Peek  4.Display  5.Exit
```

```
Enter choice: 1
Enter value: 3420

1.Push  2.Pop  3.Peek  4.Display  5.Exit
Enter choice: 1
Enter value: 2034

1.Push  2.Pop  3.Peek  4.Display  5.Exit
Enter choice: 2
Popped element: 2034

1.Push  2.Pop  3.Peek  4.Display  5.Exit
Enter choice: 3
Top element: 3420

1.Push  2.Pop  3.Peek  4.Display  5.Exit
Enter choice: 4
Stack elements: 3420 20 34

1.Push  2.Pop  3.Peek  4.Display  5.Exit
Enter choice: 5
```

Q3) Write a program to reverse a string using a stack implemented with an array.

```
#include <stdio.h>
#include <string.h>
#define MAX 100

char stack[MAX];
int top=-1;

void push(char c){
    if(top==MAX-1)return;
    stack[++top]=c;
}

char pop(){
    if(top==-1)return '\0';
    return stack[top--];
}

int main(){
    char str[MAX];
    printf("Enter string: ");
    scanf("%s",str);
    for(int i=0;i<strlen(str);i++)
        push(str[i]);
    printf("Reversed string: ");
    while(top!=-1)
        printf("%c",pop());
    printf("\n");
    return 0;
}
```

Sample Input & Output

```
Enter string: YouCanIfYou

Reversed string: uoYfInaCuoY
```

Q4) Write a program to check balanced parentheses in an expression using a stack implemented with a linked list.

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

struct node{
    char data;
    struct node *next;
};

struct node *top=NULL;
```

```
void push(char c){
   struct node *n=malloc(sizeof(struct node));
   n->data=c;
   n->next=top;
   top=n;
}

char pop(){
   if(top==NULL)return '\0';
   char c=top->data;
   struct node *t=top;
   top=top->next;
   free(t);
   return c;
}

int main(){
   char exp[100];
   int flag=0;
   printf("Enter expression: ");
   scanf("%s",exp);
   for(int i=0;exp[i]!='\0';i++){
      if(exp[i]=='('||exp[i]=='{'||exp[i]=='[')
         push(exp[i]);
      else if(exp[i]==')'||exp[i]=='}'||exp[i]==']'){
         char c=pop();
         if((exp[i]==')'&&c!='(')||(exp[i]=='}'&&c!='{')||(exp[i]==']'&&c!='[')){
            flag=1;
            break;
         }
      }
   }
   if(top!=NULL)flag=1;
   if(flag==0)printf("Balanced\n");
   else printf("Not Balanced\n");
   return 0;
}
```

Sample Input & Output

```
Enter expression: Yeah2034
```

```
Balanced
```

Q5) Implement a queue using an array and perform enqueue, dequeue, and display operations with overflow and underflow checks.

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

#define MAX 5
int queue[MAX];
int front=-1,rear=-1;

void enqueue(int x){
   if(rear==MAX-1){
      printf("Queue Overflow\n");
      return;
   }
   if(front==-1)front=0;
   queue[++rear]=x;
}

void dequeue(){
   if(front==-1||front>rear){
      printf("Queue Underflow\n");
      return;
   }
   printf("Dequeued element: %d\n",queue[front++]);
}

void display(){
   if(front==-1||front>rear){
      printf("Queue empty\n");
      return;
   }
   printf("Queue elements: ");
   for(int i=front;i<=rear;i++)
      printf("%d ",queue[i]);
   printf("\n");
}
```

```c
int main(){
   int ch,val;
   while(1){
      printf("\n1.Enqueue  2.Dequeue  3.Display  4.Exit\nEnter choice: ");
      scanf("%d",&ch);
      switch(ch){
         case 1:
            printf("Enter value: ");
            scanf("%d",&val);
            enqueue(val);
            break;
         case 2:
            dequeue();
            break;
         case 3:
            display();
            break;
         case 4:
            exit(0);
         default:
            printf("Invalid choice\n");
      }
   }
   return 0;
}
```

Sample Input & Output

```
1.Enqueue  2.Dequeue  3.Display  4.Exit
Enter choice: 1
Enter value: 20


1.Enqueue  2.Dequeue  3.Display  4.Exit
Enter choice: 1
Enter value: 34


1.Enqueue  2.Dequeue  3.Display  4.Exit
Enter choice: 1
Enter value: 2034


1.Enqueue  2.Dequeue  3.Display  4.Exit

Enter choice: 2

Dequeued element: 20


1.Enqueue  2.Dequeue  3.Display  4.Exit

Enter choice: 20

Invalid choice


1.Enqueue  2.Dequeue  3.Display  4.Exit

Enter choice: 3

Queue elements: 34 2034


1.Enqueue  2.Dequeue  3.Display  4.Exit

Enter choice: 4
```

Q6) Implement a queue using a linked list and perform enqueue, dequeue, and display operations.

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

struct node{
   int data;
   struct node *next;
};

struct node *front=NULL,*rear=NULL;

void enqueue(int x){
   struct node *n=malloc(sizeof(struct node));
   n->data=x;
   n->next=NULL;
   if(rear==NULL){
       front=rear=n;
       return;
   }
   rear->next=n;
   rear=n;
}

void dequeue(){
   if(front==NULL){
       printf("Queue Underflow\n");
       return;
   }
   printf("Dequeued element: %d\n",front->data);
   struct node *t=front;
   front=front->next;
   if(front==NULL)rear=NULL;
   free(t);
}

void display(){
   if(front==NULL){
       printf("Queue empty\n");
       return;
   }
   struct node *t=front;
   printf("Queue elements: ");
   while(t){
       printf("%d ",t->data);
       t=t->next;
   }
   printf("\n");
}

int main(){
   int ch,val;
   while(1){
       printf("\n1.Enqueue  2.Dequeue  3.Display  4.Exit\nEnter choice: ");
       scanf("%d",&ch);
       switch(ch){
           case 1:
               printf("Enter value: ");
               scanf("%d",&val);
            enqueue(val);
            break;
         case 2:
            dequeue();
            break;
         case 3:
            display();
            break;
         case 4:
            exit(0);
         default:
            printf("Invalid choice\n");
      }
   }
   return 0;
}
```

Sample Input & Output

```
1.Enqueue  2.Dequeue  3.Display  4.Exit

Enter choice: 1

Enter value: 34


1.Enqueue  2.Dequeue  3.Display  4.Exit

Enter choice: 1

Enter value: 20


1.Enqueue  2.Dequeue  3.Display  4.Exit

Enter choice: 1

Enter value: 2034


1.Enqueue  2.Dequeue  3.Display  4.Exit

Enter choice: 1

Enter value: 3420


1.Enqueue  2.Dequeue  3.Display  4.Exit

Enter choice: 2
```

```
Dequeued element: 34


1.Enqueue  2.Dequeue  3.Display  4.Exit
Enter choice: 2
Dequeued element: 20


1.Enqueue  2.Dequeue  3.Display  4.Exit
Enter choice: 3
Queue elements: 2034 3420


1.Enqueue  2.Dequeue  3.Display  4.Exit
Enter choice: 4
```

Q7) Implement a circular queue using an array and perform enqueue, dequeue, and display operations demonstrating wrap-around behavior.

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

#define MAX 5
int queue[MAX];
int front=-1,rear=-1;

void enqueue(int x){
   if((rear+1)%MAX==front){
      printf("Queue Overflow\n");
      return;
   }
   if(front==-1)front=0;
   rear=(rear+1)%MAX;
   queue[rear]=x;
}

void dequeue(){
   if(front==-1){
      printf("Queue Underflow\n");
```

```
        return;
    }
    printf("Dequeued element: %d\n",queue[front]);
    if(front==rear){
        front=rear=-1;
        return;
    }
    front=(front+1)%MAX;
}

void display(){
    if(front==-1){
        printf("Queue empty\n");
        return;
    }
    printf("Queue elements: ");
    int i=front;
    while(1){
        printf("%d ",queue[i]);
        if(i==rear)break;
        i=(i+1)%MAX;
    }
    printf("\n");
}

int main(){
    int ch,val;
    while(1){
        printf("\n1.Enqueue  2.Dequeue  3.Display  4.Exit\nEnter choice: ");
        scanf("%d",&ch);
        switch(ch){
            case 1:
                printf("Enter value: ");
                scanf("%d",&val);
                enqueue(val);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
```

```c
            case 4:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

Sample Input & Output

```
1.Enqueue  2.Dequeue  3.Display  4.Exit
Enter choice: 1
Enter value: 20

1.Enqueue  2.Dequeue  3.Display  4.Exit
Enter choice: 1
Enter value: 34

1.Enqueue  2.Dequeue  3.Display  4.Exit
Enter choice: 1
Enter value: 2034

1.Enqueue  2.Dequeue  3.Display  4.Exit
Enter choice: 1
Enter value: 3420

1.Enqueue  2.Dequeue  3.Display  4.Exit
Enter choice: 2
Dequeued element: 20

1.Enqueue  2.Dequeue  3.Display  4.Exit
Enter choice: 2
Dequeued element: 34
```

```
1.Enqueue  2.Dequeue  3.Display  4.Exit
Enter choice: 3
Queue elements: 2034 3420


1.Enqueue  2.Dequeue  3.Display  4.Exit
Enter choice: 4
```

Q8) Implement a priority queue using an array, where each element has a (data, priority) pair, and dequeue should remove the element with the highest priority.

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

#define MAX 10
int data[MAX],priority[MAX];
int size=0;

void enqueue(int val,int pr){
   if(size==MAX){
      printf("Queue Overflow\n");
      return;
   }
   int i=size-1;
   while(i>=0 && priority[i]<pr){
      data[i+1]=data[i];
      priority[i+1]=priority[i];
      i--;
   }
   data[i+1]=val;
   priority[i+1]=pr;
   size++;
}

void dequeue(){
   if(size==0){
      printf("Queue Underflow\n");
```

```
        return;
    }
    printf("Dequeued element: %d (Priority %d)\n",data[0],priority[0]);
    for(int i=0;i<size-1;i++){
        data[i]=data[i+1];
        priority[i]=priority[i+1];
    }
    size--;
}

void display(){
    if(size==0){
        printf("Queue empty\n");
        return;
    }
    printf("Priority Queue: ");
    for(int i=0;i<size;i++)
        printf("[%d|%d] ",data[i],priority[i]);
    printf("\n");
}

int main(){
    int ch,val,pr;
    while(1){
        printf("\n1.Enqueue  2.Dequeue  3.Display  4.Exit\nEnter choice: ");
        scanf("%d",&ch);
        switch(ch){
            case 1:
                printf("Enter value and priority: ");
                scanf("%d%d",&val,&pr);
                enqueue(val,pr);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

Sample Input & Output

```
1.Enqueue  2.Dequeue  3.Display  4.Exit

Enter choice: 1

Enter value and priority: 20

1


1.Enqueue  2.Dequeue  3.Display  4.Exit

Enter choice: 34  1

Enter value and priority: 34

2


1.Enqueue  2.Dequeue  3.Display  4.Exit

Enter choice: 1

Enter value and priority: 3420

1


1.Enqueue  2.Dequeue  3.Display  4.Exit

Enter choice: 4 3

Priority Queue: [34|2] [20|1] [3420|1]


1.Enqueue  2.Dequeue  3.Display  4.Exit

Enter choice: 4
```

Q9) Implement a double-ended queue (deque) using a circular array with insert and delete operations at both ends.

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

```
#define MAX 5
int deque[MAX];
int front=-1,rear=-1;

int isFull(){
   return((front==0 && rear==MAX-1)||(front==rear+1));
}

int isEmpty(){
   return(front==-1);
}

void insertFront(int x){
   if(isFull()){
      printf("Deque Overflow\n");
      return;
   }
   if(front==-1)
      front=rear=0;
   else if(front==0)
      front=MAX-1;
   else
      front--;
   deque[front]=x;
}

void insertRear(int x){
   if(isFull()){
      printf("Deque Overflow\n");
      return;
   }
   if(front==-1)
      front=rear=0;
   else if(rear==MAX-1)
      rear=0;
   else
      rear++;
   deque[rear]=x;
}

void deleteFront(){
    if(isEmpty()){
        printf("Deque Underflow\n");
        return;
    }
    printf("Deleted element: %d\n",deque[front]);
    if(front==rear)
        front=rear=-1;
    else if(front==MAX-1)
        front=0;
    else
        front++;
}

void deleteRear(){
    if(isEmpty()){
        printf("Deque Underflow\n");
        return;
    }
    printf("Deleted element: %d\n",deque[rear]);
    if(front==rear)
        front=rear=-1;
    else if(rear==0)
        rear=MAX-1;
    else
        rear--;
}

void display(){
    if(isEmpty()){
        printf("Deque empty\n");
        return;
    }
    printf("Deque elements: ");
    int i=front;
    while(1){
        printf("%d ",deque[i]);
        if(i==rear)break;
        i=(i+1)%MAX;
    }
    printf("\n");
}
```

```
int main(){
   int ch,val;
   while(1){
      printf("\n1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display
6.Exit\nEnter choice: ");
      scanf("%d",&ch);
      switch(ch){
         case 1:
            printf("Enter value: ");
            scanf("%d",&val);
            insertFront(val);
            break;
         case 2:
            printf("Enter value: ");
            scanf("%d",&val);
            insertRear(val);
            break;
         case 3:
            deleteFront();
            break;
         case 4:
            deleteRear();
            break;
         case 5:
            display();
            break;
         case 6:
            exit(0);
         default:
            printf("Invalid choice\n");
      }
   }
   return 0;
}
```

Sample Input & Output

```
1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit

Enter choice: 1

Enter value: 20
```

```
1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 2
Enter value: 34

1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 1
Enter value: 2034

1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 2
Enter value: 3420

1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 3 5
Deque elements: 2034 20 34 3420

1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 3
Deleted element: 2034

1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 4
Deleted element: 3420

1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 5
Deque elements: 20 34

1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 6
```

Q10) Write a program using both stack and queue to check whether a given string is a palindrome.

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

#define MAX 5
int deque[MAX];
int front=-1,rear=-1;

int isFull(){
   return((front==0 && rear==MAX-1)||(front==rear+1));
}

int isEmpty(){
   return(front==-1);
}

void insertFront(int x){
   if(isFull()){
      printf("Deque Overflow\n");
      return;
   }
   if(front==-1)
      front=rear=0;
   else if(front==0)
      front=MAX-1;
   else
      front--;
   deque[front]=x;
}

void insertRear(int x){
   if(isFull()){
      printf("Deque Overflow\n");
      return;
   }
   if(front==-1)
      front=rear=0;
   else if(rear==MAX-1)
      rear=0;
   else
```

```
        rear++;
    deque[rear]=x;
}

void deleteFront(){
    if(isEmpty()){
        printf("Deque Underflow\n");
        return;
    }
    printf("Deleted element: %d\n",deque[front]);
    if(front==rear)
        front=rear=-1;
    else if(front==MAX-1)
        front=0;
    else
        front++;
}

void deleteRear(){
    if(isEmpty()){
        printf("Deque Underflow\n");
        return;
    }
    printf("Deleted element: %d\n",deque[rear]);
    if(front==rear)
        front=rear=-1;
    else if(rear==0)
        rear=MAX-1;
    else
        rear--;
}

void display(){
    if(isEmpty()){
        printf("Deque empty\n");
        return;
    }
    printf("Deque elements: ");
    int i=front;
    while(1){
        printf("%d ",deque[i]);
        if(i==rear)break;
        i=(i+1)%MAX;
    }
    printf("\n");
}

int main(){
    int ch,val;
    while(1){
        printf("\n1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display
6.Exit\nEnter choice: ");
        scanf("%d",&ch);
        switch(ch){
            case 1:
                printf("Enter value: ");
                scanf("%d",&val);
                insertFront(val);
                break;
            case 2:
                printf("Enter value: ");
                scanf("%d",&val);
                insertRear(val);
                break;
            case 3:
                deleteFront();
                break;
            case 4:
                deleteRear();
                break;
            case 5:
                display();
                break;
            case 6:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

Sample Input & Output

```
1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 1
Enter value: 34

1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 1
Enter value: 3420

1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 1
Enter value: 20

1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 2
Enter value: 2034

1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 5
Deque elements: 20 3420 34 2034

1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 3
Deleted element: 20

1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 4
Deleted element: 2034

1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 5
Deque elements: 3420 34
```

```
1.InsertFront  2.InsertRear  3.DeleteFront  4.DeleteRear  5.Display  6.Exit
Enter choice: 6
```